Distributed Operating Systems

Side-Channels

Marcus Hähnel (marcus.haehnel@kernkonzept.com)

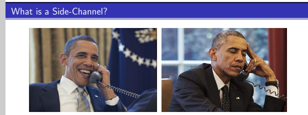
2023-07-03

2023-07-04

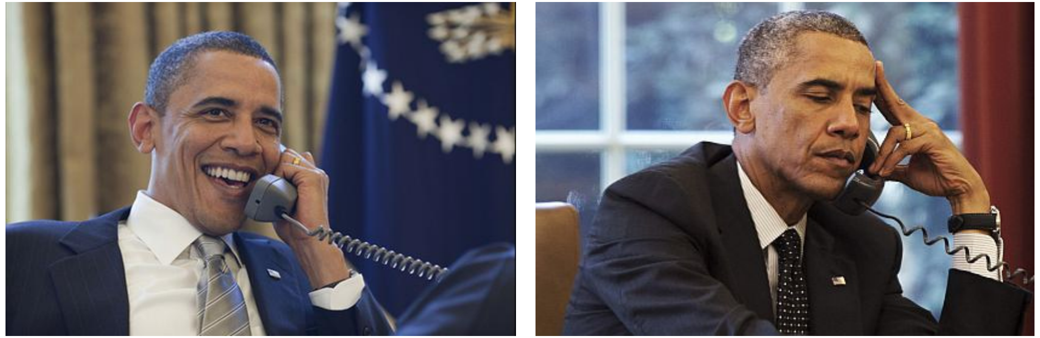
DOS - Side-Channels

Distributed Operating Systems
Side-Channels
Marcus Hähnel (marcus.haehnel@kernkonzept.com)
2023-07-03

What is a Side-Channel?



What is a Side-Channel?



Visual side-channel

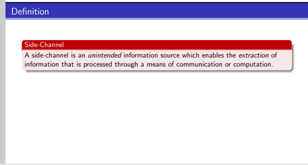
Which call has a positive connotation?

What is a Side-Channel?
Visual side-channel
Which call has a positive connotation?

Definition

Side-Channel

A side-channel is an *unintended* information source which enables the *extraction* of information that is processed through a means of communication or computation.



Definition

Side-Channel

A side-channel is an *unintended* information source which enables the *extraction* of information that is processed through a means of communication or computation.

1. The presence of the side-channel does not depend on the presence of bugs

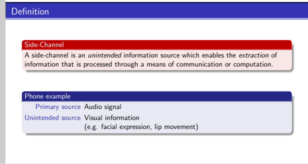
Definition

Side-Channel

A side-channel is an *unintended* information source which enables the *extraction* of information that is processed through a means of communication or computation.

Phone example

Primary source	Audio signal
Unintended source	Visual information (e.g. facial expression, lip movement)


Definition

Side-Channel
A side-channel is an *unintended* information source which enables the *extraction* of information that is processed through a means of communication or computation.

Phone example
Primary source Audio signal
Unintended source Visual information
(e.g. facial expression, lip movement)

1. The presence of the side-channel does not depend on the presence of bugs

2001: A Space Odysee — Video

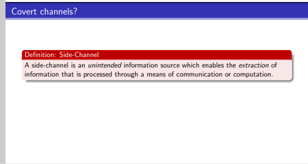
2001: A Space Odysee — Video

Covert channels?

Covert channels?

Definition: Side-Channel

A side-channel is an *unintended* information source which enables the *extraction* of information that is processed through a means of communication or computation.


Covert channels?

Definition: Side-Channel

A side-channel is an *unintended* information source which enables the *extraction* of information that is processed through a means of communication or computation.

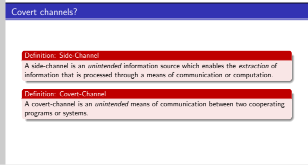
Covert channels?

Definition: Side-Channel

A side-channel is an *unintended* information source which enables the *extraction* of information that is processed through a means of communication or computation.

Definition: Covert-Channel

A covert-channel is an *unintended* means of communication between two cooperating programs or systems.


Covert channels?

Definition: Side-Channel
A side-channel is an *unintended* information source which enables the *extraction* of information that is processed through a means of communication or computation.

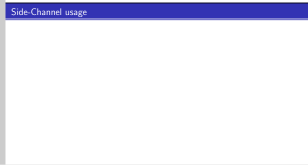
Definition: Covert-Channel
A covert-channel is an *unintended* means of communication between two cooperating programs or systems.

Side-Channel usage

2023-07-04

DOS - Side-Channels
└Introduction

└Side-Channel usage

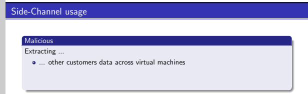

Side-Channel usage

Side-Channel usage

Malicious

Extracting ...

- ... other customers data across virtual machines


Side-Channel usage

Malicious

Extracting ...

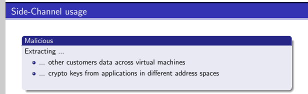
- ... other customers data across virtual machines

Side-Channel usage

Malicious

Extracting ...

- ... other customers data across virtual machines
- ... crypto keys from applications in different address spaces


Side-Channel usage

Malicious

Extracting ...

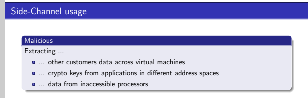
- ... other customers data across virtual machines
- ... crypto keys from applications in different address spaces

Side-Channel usage

Malicious

Extracting ...

- ... other customers data across virtual machines
- ... crypto keys from applications in different address spaces
- ... data from inaccessible processors


Side-Channel usage

Malicious

Extracting ...

- ... other customers data across virtual machines
- ... crypto keys from applications in different address spaces
- ... data from inaccessible processors

Side-Channel usage

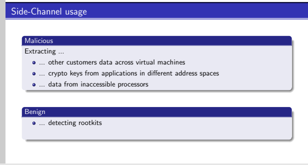
Malicious

Extracting ...

- ... other customers data across virtual machines
- ... crypto keys from applications in different address spaces
- ... data from inaccessible processors

Benign

- ... detecting rootkits


Side-Channel usage

Malicious

Extracting ...

- ... other customers data across virtual machines
- ... crypto keys from applications in different address spaces
- ... data from inaccessible processors

Benign

- ... detecting rootkits

Side-Channel usage

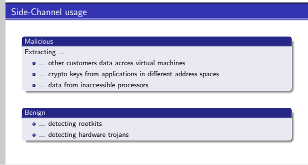
Malicious

Extracting ...

- ... other customers data across virtual machines
- ... crypto keys from applications in different address spaces
- ... data from inaccessible processors

Benign

- ... detecting rootkits
- ... detecting hardware trojans


Side-Channel usage

Malicious

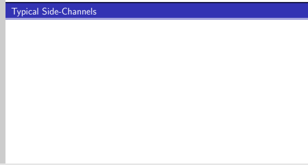
Extracting ...

- ... other customers data across virtual machines
- ... crypto keys from applications in different address spaces
- ... data from inaccessible processors

Benign

- ... detecting rootkits
- ... detecting hardware trojans

Typical Side-Channels


Typical Side-Channels

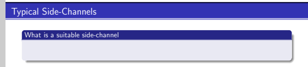
Typical Side-Channels

What is a suitable side-channel

DOS - Side-Channels
└Introduction

2023-07-04

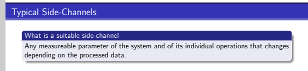
└Typical Side-Channels



Typical Side-Channels

What is a suitable side-channel

Any measureable parameter of the system and of its individual operations that changes depending on the processed data.


Typical Side-Channels
What is a suitable side-channel
Any measureable parameter of the system and of its individual operations that changes depending on the processed data.

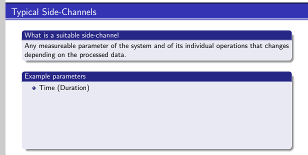
Typical Side-Channels

What is a suitable side-channel

Any measureable parameter of the system and of its individual operations that changes depending on the processed data.

Example parameters

- Time (Duration)


Typical Side-Channels
What is a suitable side-channel
Any measureable parameter of the system and of its individual operations that changes depending on the processed data.
Example parameters
• Time (Duration)

Typical Side-Channels

What is a suitable side-channel

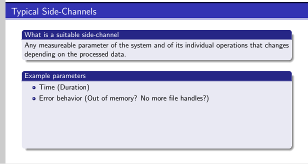
Any measureable parameter of the system and of its individual operations that changes depending on the processed data.

Example parameters

- Time (Duration)
- Error behavior (Out of memory? No more file handles?)

2023-07-04

DOS - Side-Channels
└Introduction
└Typical Side-Channels


Typical Side-Channels

What is a suitable side-channel

Any measureable parameter of the system and of its individual operations that changes depending on the processed data.

Example parameters

- Time (Duration)
- Error behavior (Out of memory? No more file handles?)

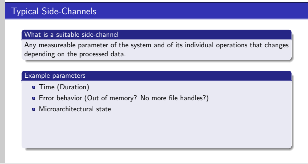
Typical Side-Channels

What is a suitable side-channel

Any measureable parameter of the system and of its individual operations that changes depending on the processed data.

Example parameters

- Time (Duration)
- Error behavior (Out of memory? No more file handles?)
- Microarchitectural state


Typical Side-Channels

What is a suitable side-channel

Any measureable parameter of the system and of its individual operations that changes depending on the processed data.

Example parameters

- Time (Duration)
- Error behavior (Out of memory? No more file handles?)
- Microarchitectural state

Typical Side-Channels

What is a suitable side-channel

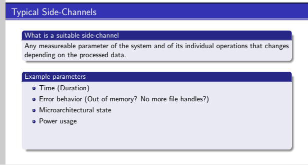
Any measureable parameter of the system and of its individual operations that changes depending on the processed data.

Example parameters

- Time (Duration)
- Error behavior (Out of memory? No more file handles?)
- Microarchitectural state
- Power usage

2023-07-04

DOS - Side-Channels
└Introduction
└Typical Side-Channels



Typical Side-Channels

What is a suitable side-channel

Any measureable parameter of the system and of its individual operations that changes depending on the processed data.

Example parameters

- Time (Duration)
- Error behavior (Out of memory? No more file handles?)
- Microarchitectural state
- Power usage

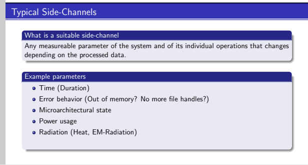
Typical Side-Channels

What is a suitable side-channel

Any measureable parameter of the system and of its individual operations that changes depending on the processed data.

Example parameters

- Time (Duration)
- Error behavior (Out of memory? No more file handles?)
- Microarchitectural state
- Power usage
- Radiation (Heat, EM-Radiation)


Typical Side-Channels

What is a suitable side-channel

Any measureable parameter of the system and of its individual operations that changes depending on the processed data.

Example parameters

- Time (Duration)
- Error behavior (Out of memory? No more file handles?)
- Microarchitectural state
- Power usage
- Radiation (Heat, EM-Radiation)

Typical Side-Channels

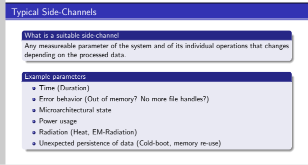
What is a suitable side-channel

Any measureable parameter of the system and of its individual operations that changes depending on the processed data.

Example parameters

- Time (Duration)
- Error behavior (Out of memory? No more file handles?)
- Microarchitectural state
- Power usage
- Radiation (Heat, EM-Radiation)
- Unexpected persistence of data (Cold-boot, memory re-use)

2023-07-04


Typical Side-Channels

What is a suitable side-channel

Any measureable parameter of the system and of its individual operations that changes depending on the processed data.

Example parameters

- Time (Duration)
- Error behavior (Out of memory? No more file handles?)
- Microarchitectural state
- Power usage
- Radiation (Heat, EM-Radiation)
- Unexpected persistence of data (Cold-boot, memory re-use)

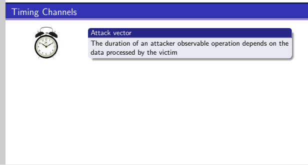
Timing Channels

Attack vector

The duration of an attacker observable operation depends on the data processed by the victim

2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
└Timing Channels
└Timing Channels

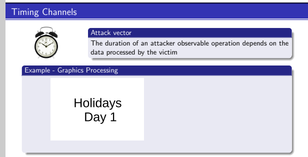


Timing Channels

Attack vector

The duration of an attacker observable operation depends on the data processed by the victim

Example - Graphics Processing



Timing Channels

Attack vector

The duration of an attacker observable operation depends on the data processed by the victim

Example - Graphics Processing

2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
 └Timing Channels
  └Timing Channels

Timing Channels

Attack vector

The duration of an attacker observable operation depends on the data processed by the victim

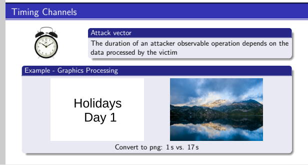
Example - Graphics Processing

Holidays
Day 1

Convert to png: 1 s vs. 17 s

2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
 └Timing Channels
  └Timing Channels



Timing Channels

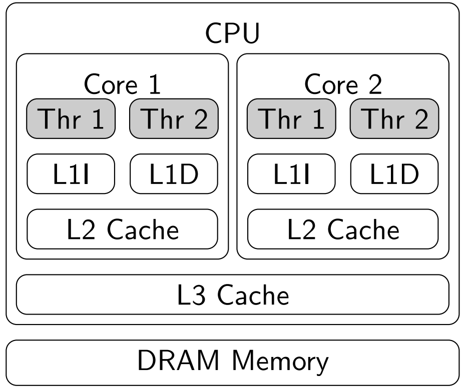
Attack vector
The duration of an attacker observable operation depends on the data processed by the victim

Example - Graphics Processing

Holidays
Day 1

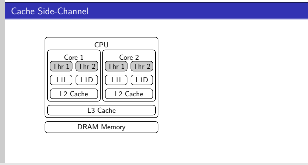
Convert to png: 1 s vs. 17 s

Cache Side-Channel

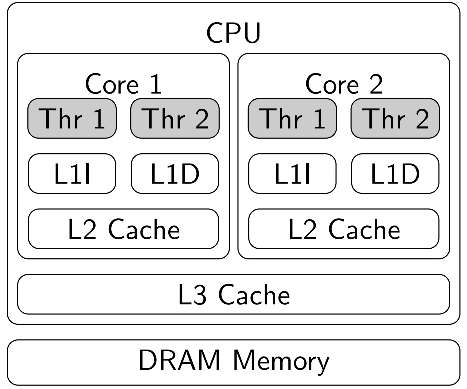


2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
  └Timing Channels
    └Cache Side-Channel



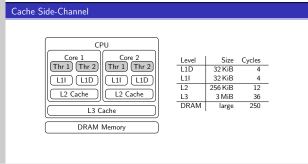
Cache Side-Channel



Level	Size	Cycles
L1D	32 KiB	4
L1I	32 KiB	4
L2	256 KiB	12
L3	3 MiB	36
DRAM	large	250

2023-07-04

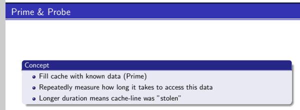
DOS - Side-Channels
└Internal Attack Vectors
└Timing Channels
└Cache Side-Channel



Prime & Probe

Concept

- Fill cache with known data (Prime)
- Repeatedly measure how long it takes to access this data
- Longer duration means cache-line was "stolen"



Prime & Probe

Concept

- Fill cache with known data (Prime)
- Repeatedly measure how long it takes to access this data
- Longer duration means cache-line was "stolen"

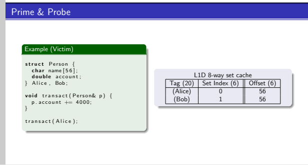
Prime & Probe

Example (Victim)

```
struct Person {
  char name[56];
  double account;
} Alice, Bob;

void transact(Person& p) {
  p.account += 4000;
}

transact(Alice);
```

L1D 8-way set cache

Tag (20)	Set Index (6)	Offset (6)
(Alice)	0	56
(Bob)	1	56



Prime & Probe

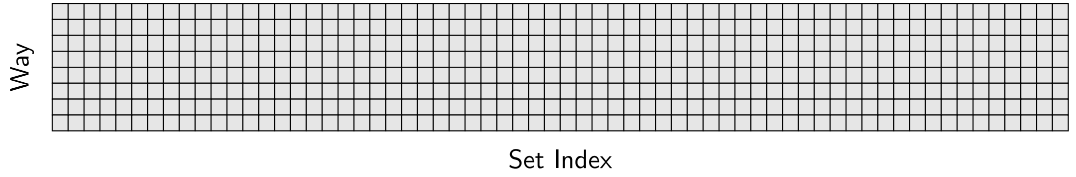
Example (Victim)

```
struct Person {
  char name[56];
  double account;
} Alice, Bob;
```

L1D 8-way set cache

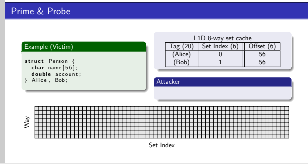
Tag (20)	Set Index (6)	Offset (6)
(Alice)	0	56
(Bob)	1	56

Attacker



2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
  └Timing Channels
    └Prime & Probe



Prime & Probe

Example (Victim)

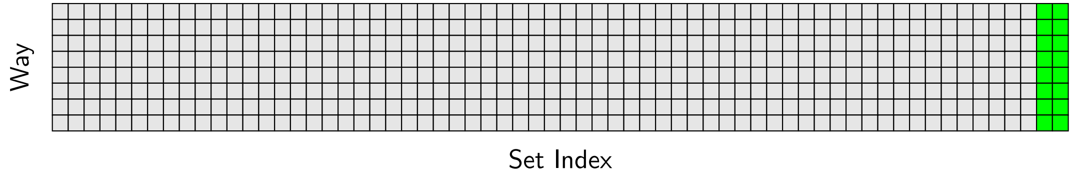
```
struct Person {
  char name[56];
  double account;
} Alice, Bob;
```

L1D 8-way set cache

Tag (20)	Set Index (6)	Offset (6)
(Alice)	0	56
(Bob)	1	56

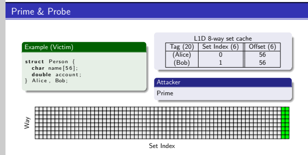
Attacker

Prime



2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
 └Timing Channels
  └Prime & Probe



Prime & Probe

Example (Victim)

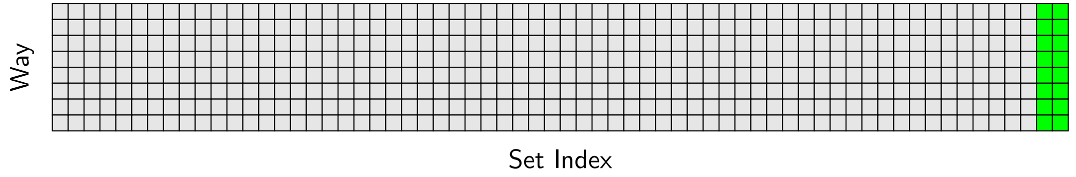
```
struct Person {
  char name[56];
  double account;
} Alice, Bob;
```

L1D 8-way set cache

Tag (20)	Set Index (6)	Offset (6)
(Alice)	0	56
(Bob)	1	56

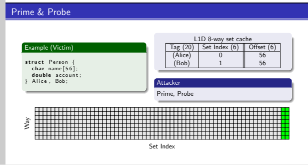
Attacker

Prime, Probe



2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
  └Timing Channels
    └Prime & Probe


Prime & Probe
Example (Victim)
struct Person { char name[56]; double account; } Alice, Bob;
L1D 8-way set cache
Tag (20) | Set Index (6) | Offset (6)
(Alice) | 0 | 56
(Bob) | 1 | 56
Attacker
Prime, Probe
Way
Set Index

Prime & Probe

Example (Victim)

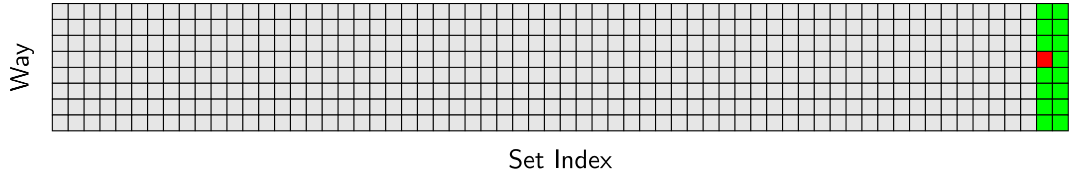
```
struct Person {
  char name[56];
  double account;
} Alice, Bob;
```

L1D 8-way set cache

Tag (20)	Set Index (6)	Offset (6)
(Alice)	0	56
(Bob)	1	56

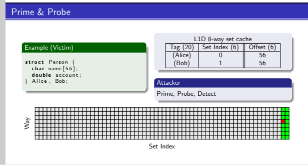
Attacker

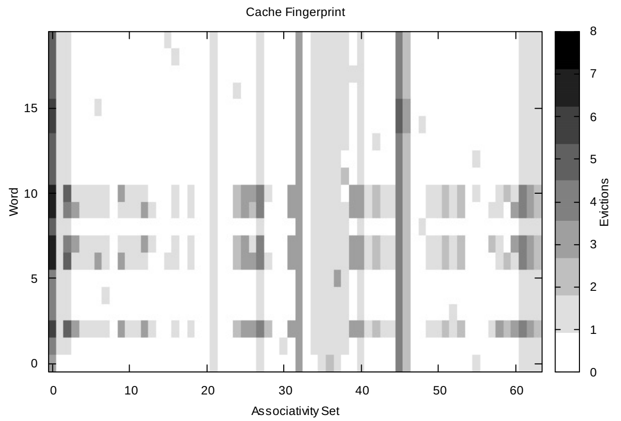
Prime, Probe, Detect



2023-07-04

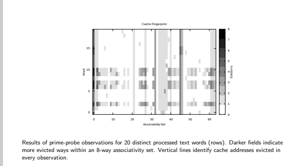
DOS - Side-Channels
└Internal Attack Vectors
  └Timing Channels
    └Prime & Probe





Results of prime-probe observations for 20 distinct processed text words (rows). Darker fields indicate more evicted ways within an 8-way associativity set. Vertical lines identify cache addresses evicted in every observation.

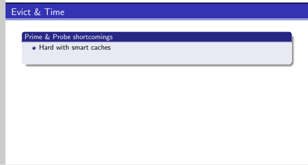
DOS - Side-Channels
└Internal Attack Vectors
└Timing Channels



Evict & Time

Prime & Probe shortcomings

- Hard with smart caches


Evict & Time

Prime & Probe shortcomings

- Hard with smart caches

Evict & Time

Prime & Probe shortcomings

- Hard with smart caches
- Probing is prone to many false positives

2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
└Timing Channels
└Evict & Time

Evict & Time

Prime & Probe shortcomings
- Hard with smart caches
- Probing is prone to many false positives

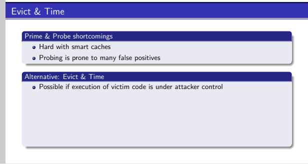
Evict & Time

Prime & Probe shortcomings

- Hard with smart caches
- Probing is prone to many false positives

Alternative: Evict & Time

- Possible if execution of victim code is under attacker control


Evict & Time
Prime & Probe shortcomings
- Hard with smart caches
- Probing is prone to many false positives
Alternative: Evict & Time
- Possible if execution of victim code is under attacker control

Evict & Time

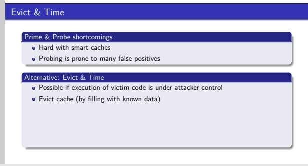
Prime & Probe shortcomings

- Hard with smart caches
- Probing is prone to many false positives

Alternative: Evict & Time

- Possible if execution of victim code is under attacker control
- Evict cache (by filling with known data)

DOS - Side-Channels
└Internal Attack Vectors
  └Timing Channels
    └Evict & Time


Evict & Time

Prime & Probe shortcomings
- Hard with smart caches
- Probing is prone to many false positives

Alternative: Evict & Time
- Possible if execution of victim code is under attacker control
- Evict cache (by filling with known data)

Evict & Time

Prime & Probe shortcomings

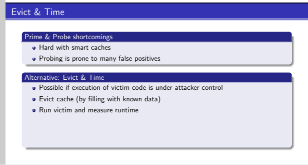
- Hard with smart caches
- Probing is prone to many false positives

Alternative: Evict & Time

- Possible if execution of victim code is under attacker control
- Evict cache (by filling with known data)
- Run victim and measure runtime

2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
  └Timing Channels
    └Evict & Time


Evict & Time

Prime & Probe shortcomings
- Hard with smart caches
- Probing is prone to many false positives

Alternative: Evict & Time
- Possible if execution of victim code is under attacker control
- Evict cache (by filling with known data)
- Run victim and measure runtime

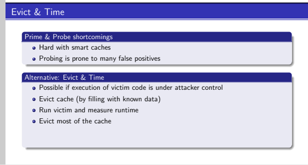
Evict & Time

Prime & Probe shortcomings

- Hard with smart caches
- Probing is prone to many false positives

Alternative: Evict & Time

- Possible if execution of victim code is under attacker control
- Evict cache (by filling with known data)
- Run victim and measure runtime
- Evict most of the cache


Evict & Time

Prime & Probe shortcomings
- Hard with smart caches
- Probing is prone to many false positives

Alternative: Evict & Time
- Possible if execution of victim code is under attacker control
- Evict cache (by filling with known data)
- Run victim and measure runtime
- Evict most of the cache

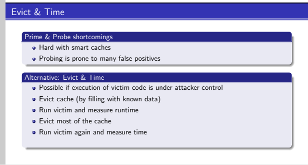
Evict & Time

Prime & Probe shortcomings

- Hard with smart caches
- Probing is prone to many false positives

Alternative: Evict & Time

- Possible if execution of victim code is under attacker control
- Evict cache (by filling with known data)
- Run victim and measure runtime
- Evict most of the cache
- Run victim again and measure time



Evict & Time

Prime & Probe shortcomings

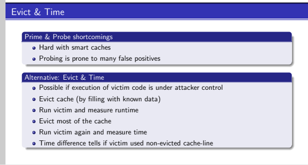
- Hard with smart caches
- Probing is prone to many false positives

Alternative: Evict & Time

- Possible if execution of victim code is under attacker control
- Evict cache (by filling with known data)
- Run victim and measure runtime
- Evict most of the cache
- Run victim again and measure time
- Time difference tells if victim used non-evicted cache-line

2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
 └Timing Channels
  └Evict & Time


Evict & Time

Prime & Probe shortcomings
- Hard with smart caches
- Probing is prone to many false positives

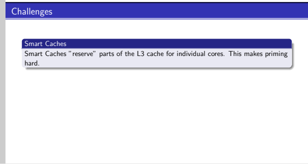
Alternative: Evict & Time
- Possible if execution of victim code is under attacker control
- Evict cache (by filling with known data)
- Run victim and measure runtime
- Evict most of the cache
- Run victim again and measure time
- Time difference tells if victim used non-evicted cache-line

Challenges

Smart Caches

Smart Caches "reserve" parts of the L3 cache for individual cores. This makes priming hard.

DOS - Side-Channels
└Internal Attack Vectors
  └Timing Channels
    └Challenges


Challenges

Smart Caches

Smart Caches "reserve" parts of the L3 cache for individual cores. This makes priming hard.

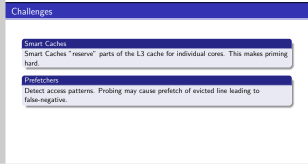
Challenges

Smart Caches

Smart Caches "reserve" parts of the L3 cache for individual cores. This makes priming hard.

Prefetchers

Detect access patterns. Probing may cause prefetch of evicted line leading to false-negative.



Challenges

Smart Caches

Smart Caches "reserve" parts of the L3 cache for individual cores. This makes priming hard.

Prefetchers

Detect access patterns. Probing may cause prefetch of evicted line leading to false-negative.

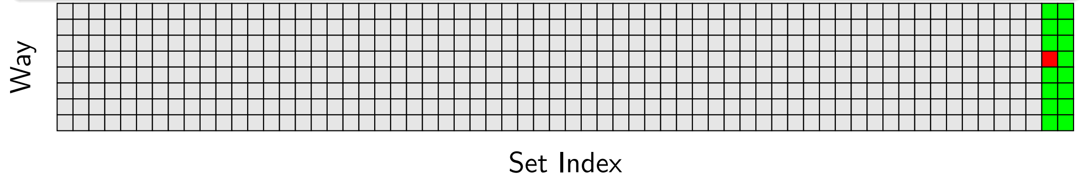
Challenges

Smart Caches

Smart Caches "reserve" parts of the L3 cache for individual cores. This makes priming hard.

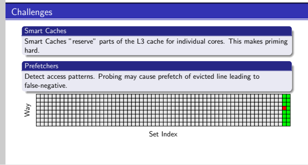
Prefetchers

Detect access patterns. Probing may cause prefetch of evicted line leading to false-negative.



2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
  └Timing Channels
    └Challenges



Challenges

Smart Caches

Smart Caches "reserve" parts of the L3 cache for individual cores. This makes priming hard.

Prefetchers

Detect access patterns. Probing may cause prefetch of evicted line leading to false-negative.

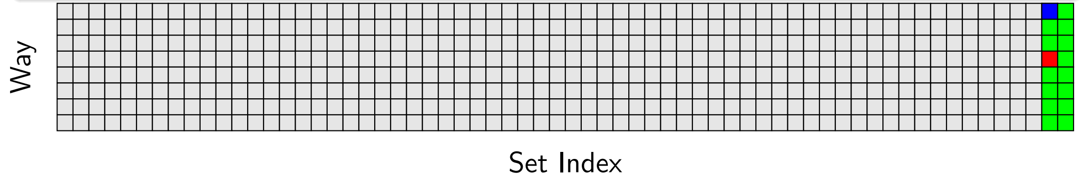
Challenges

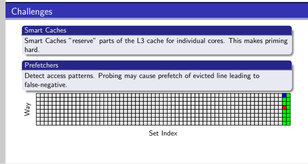
Smart Caches

Smart Caches "reserve" parts of the L3 cache for individual cores. This makes priming hard.

Prefetchers

Detect access patterns. Probing may cause prefetch of evicted line leading to false-negative.




Challenges

Smart Caches

Smart Caches "reserve" parts of the L3 cache for individual cores. This makes priming hard.

Prefetchers

Detect access patterns. Probing may cause prefetch of evicted line leading to false-negative.

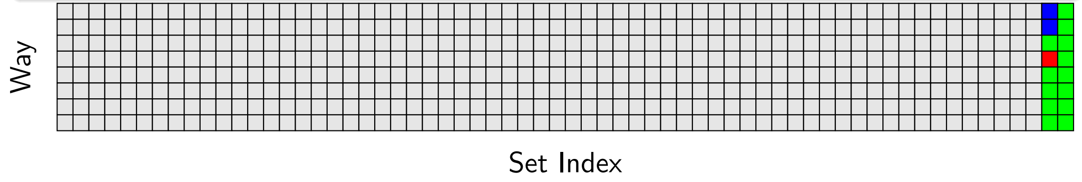
Challenges

Smart Caches

Smart Caches "reserve" parts of the L3 cache for individual cores. This makes priming hard.

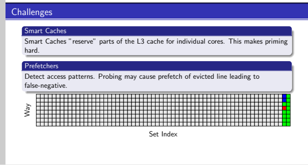
Prefetchers

Detect access patterns. Probing may cause prefetch of evicted line leading to false-negative.



2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
 └Timing Channels
  └Challenges



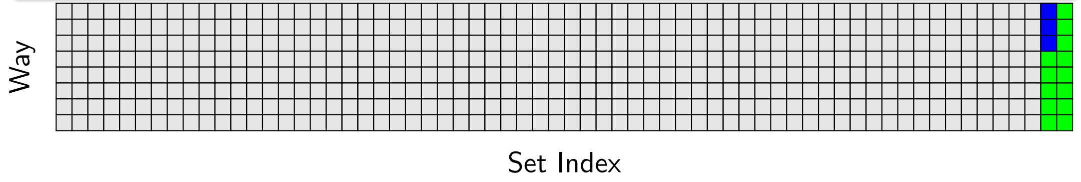
Challenges

Smart Caches

Smart Caches "reserve" parts of the L3 cache for individual cores. This makes priming hard.

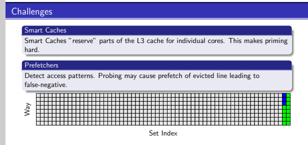
Prefetchers

Detect access patterns. Probing may cause prefetch of evicted line leading to false-negative.



2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
  └Timing Channels
    └Challenges



Challenges

Smart Caches

Smart Caches "reserve" parts of the L3 cache for individual cores. This makes priming hard.

Prefetchers

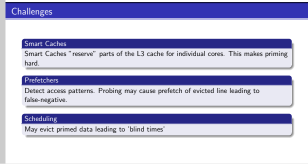
Detect access patterns. Probing may cause prefetch of evicted line leading to false-negative.

Scheduling

May evict primed data leading to 'blind times'

2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
  └Timing Channels
    └Challenges


Challenges

Smart Caches
Smart Caches "reserve" parts of the L3 cache for individual cores. This makes priming hard.

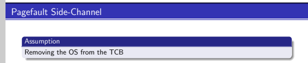
Prefetchers
Detect access patterns. Probing may cause prefetch of evicted line leading to false-negative.

Scheduling
May evict primed data leading to 'blind times'

Pagefault Side-Channel

Assumption

Removing the OS from the TCB


Pagefault Side-Channel
Assumption
Removing the OS from the TCB

Pagefault Side-Channel

Assumption

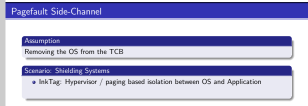
Removing the OS from the TCB

Scenario: Shielding Systems

- InkTag: Hypervisor / paging based isolation between OS and Application

2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
  └Fault Channels
    └Pagefault Side-Channel


Pagefault Side-Channel

Assumption

Removing the OS from the TCB

Scenario: Shielding Systems

- InkTag: Hypervisor / paging based isolation between OS and Application

Pagefault Side-Channel

Assumption

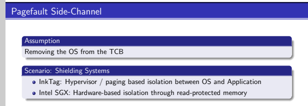
Removing the OS from the TCB

Scenario: Shielding Systems

- InkTag: Hypervisor / paging based isolation between OS and Application
- Intel SGX: Hardware-based isolation through read-protected memory

2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
  └Fault Channels
    └Pagefault Side-Channel


Pagefault Side-Channel

Assumption

Removing the OS from the TCB

Scenario: Shielding Systems

- InkTag: Hypervisor / paging based isolation between OS and Application
- Intel SGX: Hardware-based isolation through read-protected memory

Pagefault Side-Channel

Assumption

Removing the OS from the TCB

Scenario: Shielding Systems

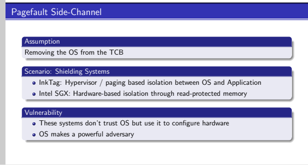
- InkTag: Hypervisor / paging based isolation between OS and Application
- Intel SGX: Hardware-based isolation through read-protected memory

Vulnerability

- These systems don't trust OS but use it to configure hardware
- OS makes a powerful adversary

2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
  └Fault Channels
    └Pagefault Side-Channel


Pagefault Side-Channel

Assumption
Removing the OS from the TCB

Scenario: Shielding Systems
- InkTag: Hypervisor / paging based isolation between OS and Application
- Intel SGX: Hardware-based isolation through read-protected memory

Vulnerability
- These systems don't trust OS but use it to configure hardware
- OS makes a powerful adversary

Controlled Channel Attacks

First attack vector against Intel SGX

Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems

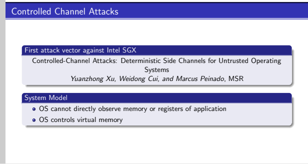
Yuanzhong Xu, Weidong Cui, and Marcus Peinado, MSR

System Model

- OS cannot directly observe memory or registers of application
- OS controls virtual memory

2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
  └Fault Channels
    └Controlled Channel Attacks


Controlled Channel Attacks

First attack vector against Intel SGX

Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems

Yuanzhong Xu, Weidong Cui, and Marcus Peinado, MSR

System Model

- OS cannot directly observe memory or registers of application
- OS controls virtual memory

Example: string length

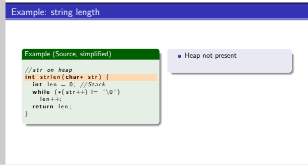
Example (Source, simplified)

```
//str on heap
int strlen(char* str) {
  int len = 0; //Stack
  while (*(str++) != '\0')
    len++;
  return len;
}
```

- Heap not present

2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
  └Fault Channels
    └Example: string length


Example: string length

Example (Source, simplified)

```
//str on heap
int strlen(char* str) {
  int len = 0; //Stack
  while (*(str++) != '\0')
    len++;
  return len;
}
```

- Heap not present

Example: string length

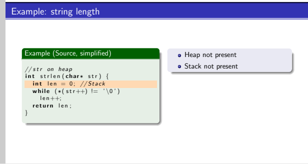
Example (Source, simplified)

```
//str on heap
int strlen(char* str) {
  int len = 0; //Stack
  while (*(str++) != '\0')
    len++;
  return len;
}
```

- Heap not present
- Stack not present

2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
  └Fault Channels
    └Example: string length


Example: string length

Example (Source, simplified)

```
//str on heap
int strlen(char* str) {
  int len = 0; //Stack
  while (*(str++) != '\0')
    len++;
  return len;
}
```

- Heap not present
- Stack not present

Example: string length

Example (Source, simplified)

```
//str on heap
int strlen(char* str) {
  int len = 0; //Stack
  while (*(str++) != '\0')
    len++;
  return len;
}
```

- Heap not present
- Stack not present

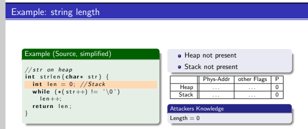
	Phys-Addr	other Flags	P
Heap	...	...	0
Stack	...	...	0

Attackers Knowledge

Length = 0

2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
  └Fault Channels
    └Example: string length


Example: string length

Example (Source, simplified)

```
//str on heap
int strlen(char* str) {
  int len = 0; //Stack
  while (*(str++) != '\0')
    len++;
  return len;
}
```

- Heap not present
- Stack not present

	Phys-Addr	other Flags	P
Heap	...	...	0
Stack	...	...	0

Attackers Knowledge

Length = 0

Example: string length

Example (Source, simplified)

```
//str on heap
int strlen(char* str) {
  int len = 0; //Stack
  while (*(str++) != '\0')
    len++;
  return len;
}
```

- Heap not present
- Stack not present

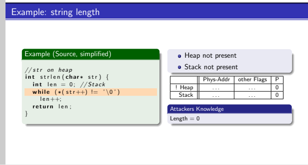
	Phys-Addr	other Flags	P
! Heap	...	...	0
Stack	...	...	0

Attackers Knowledge

Length = 0

2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
  └Fault Channels
    └Example: string length


Example: string length
Example (Source, simplified)
//str on heap
int strlen(char* str) {
int len = 0; //Stack
while (*(str++) != '\0')
len++;
return len;
}
Heap not present
Stack not present
Phys-Addr other Flags P
! Heap 0
Stack 0
Attackers Knowledge
Length = 0

Example: string length

Example (Source, simplified)

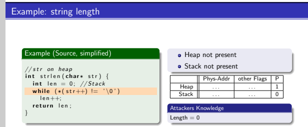
```
//str on heap
int strlen(char* str) {
  int len = 0; //Stack
  while (*(str++) != '\0')
    len++;
  return len;
}
```

- Heap not present
- Stack not present

	Phys-Addr	other Flags	P
Heap	...	...	1
Stack	...	...	0

Attackers Knowledge

Length = 0



Example: string length

Example (Source, simplified)

```
//str on heap
int strlen(char* str) {
  int len = 0; //Stack
  while (*(str++) != '\0')
    len++;
  return len;
}
```

- Heap not present
- Stack not present

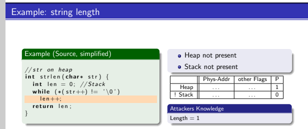
	Phys-Addr	other Flags	P
Heap	...	...	1
! Stack	...	...	0

Attackers Knowledge

Length = 1

2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
  └Fault Channels
    └Example: string length


Example: string length

Example (Source, simplified)

```
//str on heap
int strlen(char* str) {
  int len = 0; //Stack
  while (*(str++) != '\0')
    len++;
  return len;
}
```

- Heap not present
- Stack not present

	Phys-Addr	other Flags	P
Heap	...	...	1
! Stack	...	...	0

Attackers Knowledge

Length = 1

Example: string length

Example (Source, simplified)

```
//str on heap
int strlen(char* str) {
  int len = 0; //Stack
  while (*(str++) != '\0')
    len++;
  return len;
}
```

- Heap not present
- Stack not present

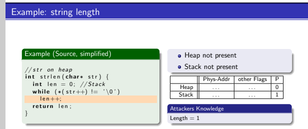
	Phys-Addr	other Flags	P
Heap	...	...	0
Stack	...	...	1

Attackers Knowledge

Length = 1

2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
  └Fault Channels
    └Example: string length


Example: string length

Example (Source, simplified)

//str on heap
int strlen(char* str) {
  int len = 0; //Stack
  while (*(str++) != '\0')
    len++;
  return len;
}

Heap not present
Stack not present

Phys-Addr | other Flags | P
Heap | ... | ... | 0
Stack | ... | ... | 1

Attackers Knowledge
Length = 1

Example: string length

Example (Source, simplified)

```
//str on heap
int strlen(char* str) {
  int len = 0; //Stack
  while (*(str++) != '\0')
    len++;
  return len;
}
```

- Heap not present
- Stack not present

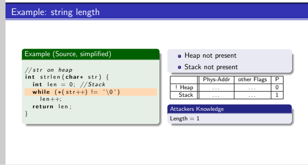
	Phys-Addr	other Flags	P
! Heap	...	...	0
Stack	...	...	1

Attackers Knowledge

Length = 1

2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
  └Fault Channels
    └Example: string length


Example: string length

Example (Source, simplified)

```
//str on heap
int strlen(char* str) {
  int len = 0; //Stack
  while (*(str++) != '\0')
    len++;
  return len;
}
```

- Heap not present
- Stack not present

	Phys-Addr	other Flags	P
! Heap	...	...	0
Stack	...	...	1

Attackers Knowledge

Length = 1

Example: string length

Example (Source, simplified)

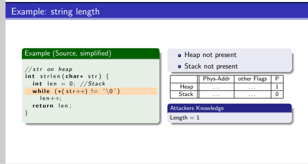
```
//str on heap
int strlen(char* str) {
  int len = 0; //Stack
  while (*(str++) != '\0')
    len++;
  return len;
}
```

- Heap not present
- Stack not present

	Phys-Addr	other Flags	P
Heap	...	...	1
Stack	...	...	0

Attackers Knowledge

Length = 1


Example: string length

Example: string length

Example (Source, simplified)

```
//str on heap
int strlen(char* str) {
  int len = 0; //Stack
  while (*(str++) != '\0')
    len++;
  return len;
}
```

- Heap not present
- Stack not present

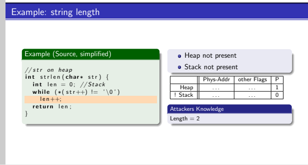
	Phys-Addr	other Flags	P
Heap	...	...	1
! Stack	...	...	0

Attackers Knowledge

Length = 2

2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
  └Fault Channels
    └Example: string length


Example: string length

Example (Source, simplified)

```
//str on heap
int strlen(char* str) {
  int len = 0; //Stack
  while (*(str++) != '\0')
    len++;
  return len;
}
```

- Heap not present
- Stack not present

	Phys-Addr	other Flags	P
Heap	...	...	1
! Stack	...	...	0

Attackers Knowledge

Length = 2

Example: string length

Example (Source, simplified)

```
//str on heap
int strlen(char* str) {
  int len = 0; //Stack
  while (*(str++) != '\0')
    len++;
  return len;
}
```

- Heap not present
- Stack not present

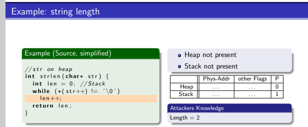
	Phys-Addr	other Flags	P
Heap	...	...	0
Stack	...	...	1

Attackers Knowledge

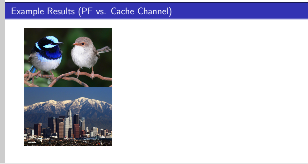
Length = 2

2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
 └Fault Channels
  └Example: string length

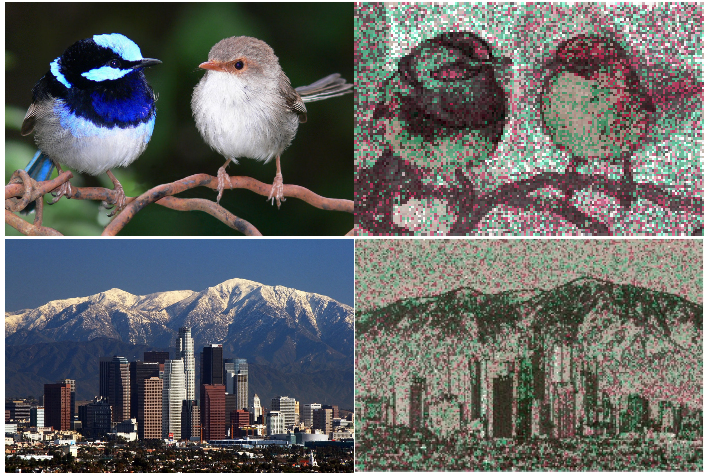


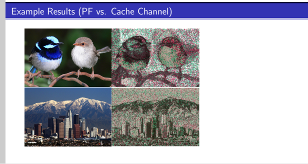
Example Results (PF vs. Cache Channel)



1. IDCT (inverse discrete cosine transformation)
2. Index in array ≈8 kB big

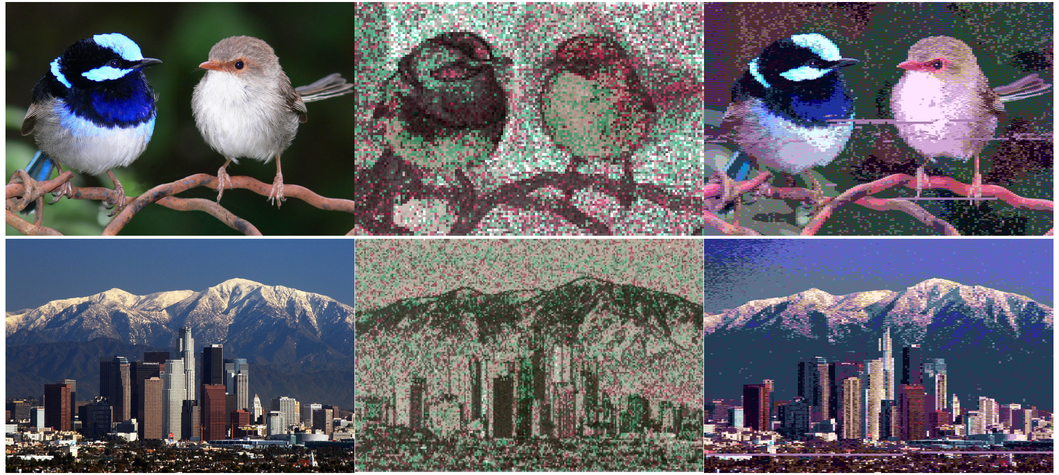
Example Results (PF vs. Cache Channel)

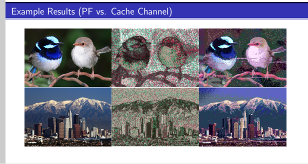




1. IDCT (inverse discrete cosine transformation)
2. Index in array ≈8 kB big

Example Results (PF vs. Cache Channel)





1. IDCT (inverse discrete cosine transformation)
2. Index in array ≈8 kB big

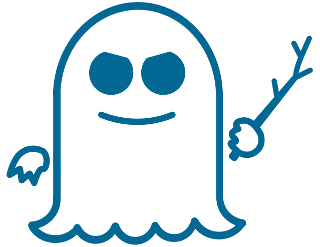
Microarchitectural Channels

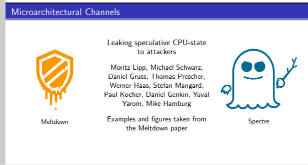
Meltdown

Leaking speculative CPU-state to attackers

Moritz Lipp, Michael Schwarz, Daniel Gruss, Thomas Prescher, Werner Haas, Stefan Mangard, Paul Kocher, Daniel Genkin, Yuval Yarom, Mike Hamburg

Examples and figures taken from the Meltdown paper


Spectre



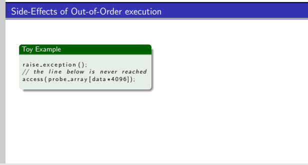
Side-Effects of Out-of-Order execution

Toy Example

```
raise_exception();
// the line below is never reached
access(probe_array[data*4096]);
```

2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
  └Microarchitectural Channels
    └Side-Effects of Out-of-Order execution


Side-Effects of Out-of-Order execution

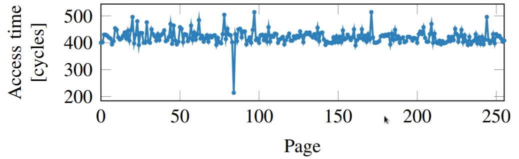
Toy Example

```
raise_exception();
// the line below is never reached
access(probe_array[data*4096]);
```

Side-Effects of Out-of-Order execution

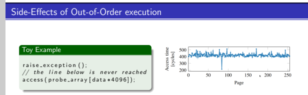
Toy Example

```
raise_exception();
// the line below is never reached
access(probe_array[data*4096]);
```



2023-07-04

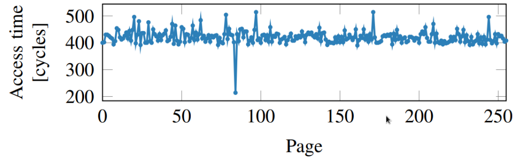
DOS - Side-Channels
└Internal Attack Vectors
└Microarchitectural Channels
└Side-Effects of Out-of-Order execution



Side-Effects of Out-of-Order execution

Toy Example

```
raise_exception();
// the line below is never reached
access(probe_array[data*4096]);
```

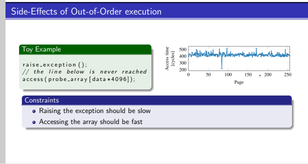


Constraints

- Raising the exception should be slow
- Accessing the array should be fast

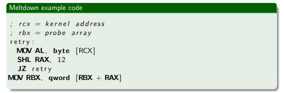
2023-07-04

DOS - Side-Channels
└Internal Attack Vectors
  └Microarchitectural Channels
    └Side-Effects of Out-of-Order execution



Meltdown example code

```
; rcx = kernel address
; rbx = probe array
retry:
  MOV AL, byte [RCX]
  SHL RAX, 12
  JZ retry
MOV RBX, qword [RBX + RAX]
```



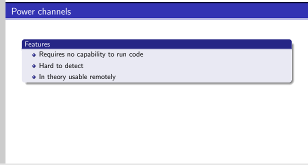
1. Retry needed because execption handling zeroes registers
2. No evicted cache line is considered zero
3. Exception can be prevented (amongst others) using TSX

Power channels

Features

- Requires no capability to run code
- Hard to detect
- In theory usable remotely

DOS - Side-Channels
└External Attack Vectors
  └Power channels
    └Power channels



Power channels

Features
- Requires no capability to run code
- Hard to detect
- In theory usable remotely

Power channels

Features

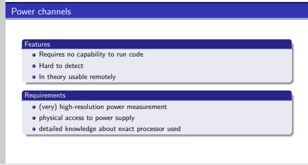
- Requires no capability to run code
- Hard to detect
- In theory usable remotely

Requirements

- (very) high-resolution power measurement
- physical access to power supply
- detailed knowledge about exact processor used

2023-07-04

DOS - Side-Channels
└External Attack Vectors
 └Power channels
  └Power channels



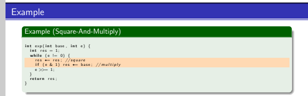
Example

Example (Square-And-Multiply)

```
int exp(int base, int e) {
  int res = 1;
  while (e != 0) {
    res *= res; //square
    if (e & 1) res *= base; //multiply
    e >>= 1;
  }
  return res;
}
```

2023-07-04

DOS - Side-Channels
└External Attack Vectors
  └Power channels
    └Example


Example

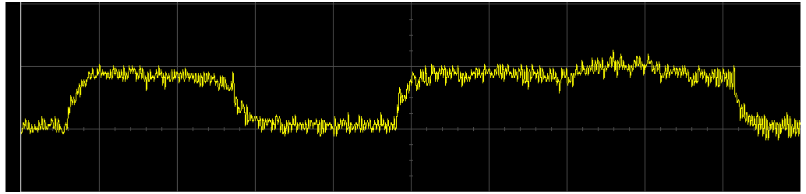
Example (Square-And-Multiply)

```
int exp(int base, int e) {
  int res = 1;
  while (e != 0) {
    res *= res; //square
    if (e & 1) res *= base; //multiply
    e >>= 1;
  }
  return res;
}
```

Example

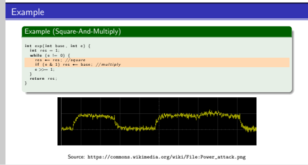
Example (Square-And-Multiply)

```
int exp(int base, int e) {
  int res = 1;
  while (e != 0) {
    res *= res; //square
    if (e & 1) res *= base; //multiply
    e >>= 1;
  }
  return res;
}
```



Source: https://commons.wikimedia.org/wiki/File:Power_attack.png

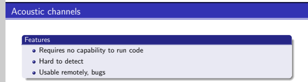
DOS - Side-Channels
└External Attack Vectors
  └Power channels
    └Example



Acoustic channels

Features

- Requires no capability to run code
- Hard to detect
- Usable remotely, bugs


Acoustic channels

Features

- Requires no capability to run code
- Hard to detect
- Usable remotely, bugs

Acoustic channels

Features

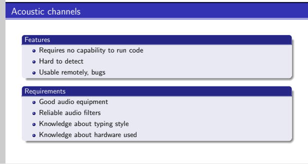
- Requires no capability to run code
- Hard to detect
- Usable remotely, bugs

Requirements

- Good audio equipment
- Reliable audio filters
- Knowledge about typing style
- Knowledge about hardware used

2023-07-04

DOS - Side-Channels
└External Attack Vectors
  └Acoustic and Radiation
    └Acoustic channels


Acoustic channels

Features
- Requires no capability to run code
- Hard to detect
- Usable remotely, bugs

Requirements
- Good audio equipment
- Reliable audio filters
- Knowledge about typing style
- Knowledge about hardware used

Example

Password typing attack

Keyboard Acoustic Emanations Revisited
*L*i Zhuang, Feng Zhou, J. D. Tygar
University of California, Berkeley

2023-07-04

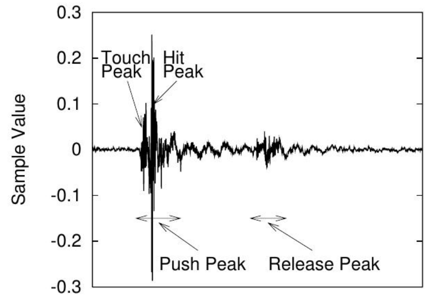
DOS - Side-Channels
└External Attack Vectors
└Acoustic and Radiation
└Example

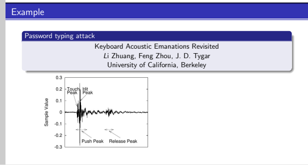
Example
Password typing attack
Keyboard Acoustic Emanations Revisited
Li Zhuang, Feng Zhou, J. D. Tygar
University of California, Berkeley

Example

Password typing attack

Keyboard Acoustic Emanations Revisited
*L*i Zhuang, Feng Zhou, J. D. Tygar
University of California, Berkeley



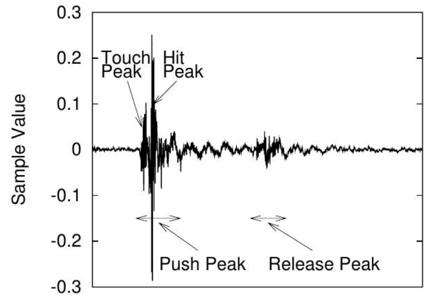


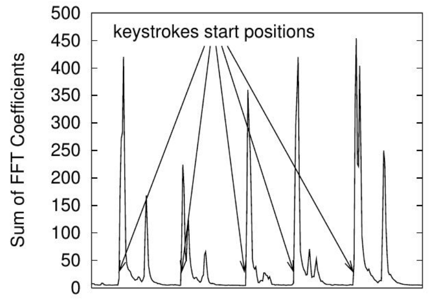
Example
Password typing attack
Keyboard Acoustic Emanations Revisited
Li Zhuang, Feng Zhou, J. D. Tygar
University of California, Berkeley

Example

Password typing attack

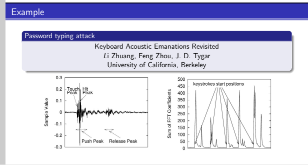
Keyboard Acoustic Emanations Revisited
*L*i Zhuang, Feng Zhou, J. D. Tygar
University of California, Berkeley



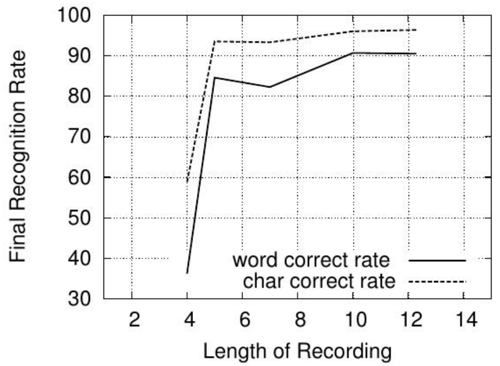


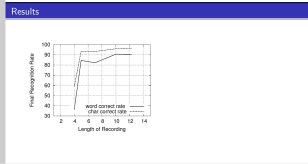
2023-07-04

DOS - Side-Channels
└External Attack Vectors
  └Acoustic and Radiation
    └Example

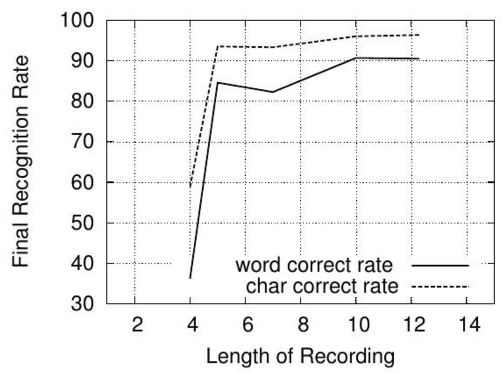


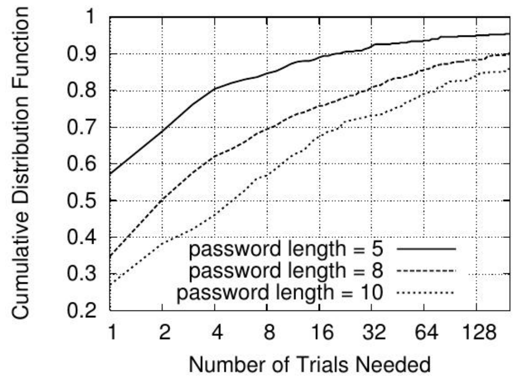
Results

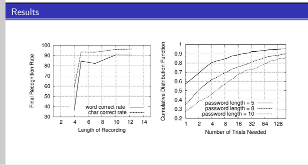

Final Recognition Rate
Length of Recording
word correct rate
char correct rate



Results


Final Recognition Rate
Length of Recording
word correct rate
char correct rate


Cumulative Distribution Function
Number of Trials Needed
password length = 5
password length = 8
password length = 10



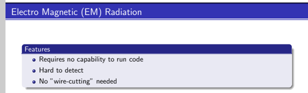
Electro Magnetic (EM) Radiation

Features

- Requires no capability to run code
- Hard to detect
- No "wire-cutting" needed

2023-07-04

DOS - Side-Channels
└External Attack Vectors
  └Acoustic and Radiation
    └Electro Magnetic (EM) Radiation


Electro Magnetic (EM) Radiation

Features

- Requires no capability to run code
- Hard to detect
- No "wire-cutting" needed

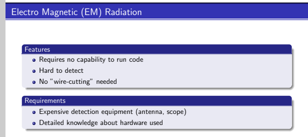
Electro Magnetic (EM) Radiation

Features

- Requires no capability to run code
- Hard to detect
- No "wire-cutting" needed

Requirements

- Expensive detection equipment (antenna, scope)
- Detailed knowledge about hardware used



Electro Magnetic (EM) Radiation

Features
- Requires no capability to run code
- Hard to detect
- No "wire-cutting" needed

Requirements
- Expensive detection equipment (antenna, scope)
- Detailed knowledge about hardware used

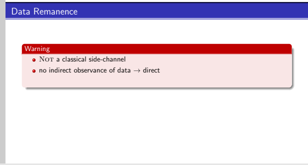
Data Remanence

Warning

- NOT a classical side-channel
- no indirect observance of data → direct

2023-07-04

DOS - Side-Channels
└Data remanence
└Data Remanence

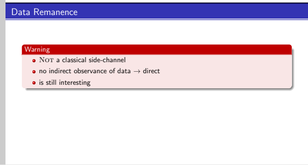

Data Remanence

Warning
- NOT a classical side-channel
- no indirect observance of data → direct

Data Remanence

Warning

- Nᴏᴛ a classical side-channel
- no indirect observance of data $\rightarrow$ direct
- is still interesting



Data Remanence

Warning

- Nᴏᴛ a classical side-channel
- no indirect observance of data $\rightarrow$ direct
- is still interesting

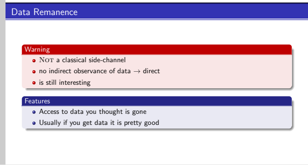
Data Remanence

Warning

- Not a classical side-channel
- no indirect observance of data → direct
- is still interesting

Features

- Access to data you thought is gone
- Usually if you get data it is pretty good


Data Remanence
Warning
Not a classical side-channel
no indirect observance of data → direct
is still interesting
Features
Access to data you thought is gone
Usually if you get data it is pretty good

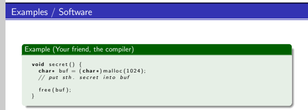
Examples / Software

Example (Your friend, the compiler)

```
void secret() {
  char* buf = (char*)malloc(1024);
  // put sth. secret into buf

  free(buf);
}
```

2023-07-04

DOS - Side-Channels
└Data remanence
  └Software
    └Examples / Software


Examples / Software

Example (Your friend, the compiler)

```
void secret() {
  char* buf = (char*)malloc(1024);
  // put sth. secret into buf

  free(buf);
}
```

Examples / Software

Example (Your friend, the compiler)

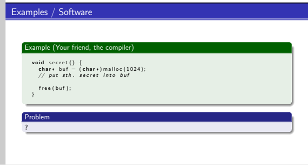
```
void secret() {
  char* buf = (char*)malloc(1024);
  // put sth. secret into buf

  free(buf);
}
```

Problem

?

2023-07-04

DOS - Side-Channels
└Data remanence
  └Software
    └Examples / Software


Examples / Software

Example (Your friend, the compiler)

void secret() {
char* buf = (char*)malloc(1024);
// put sth. secret into buf

free(buf);
}

Problem
?

Examples / Software

Example (Your friend, the compiler)

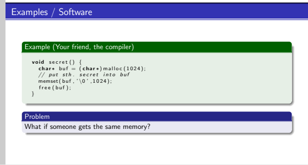
```
void secret() {
  char* buf = (char*)malloc(1024);
  // put sth. secret into buf
  memset(buf,'\0',1024);
  free(buf);
}
```

Problem

What if someone gets the same memory?

2023-07-04

DOS - Side-Channels
└Data remanence
  └Software
    └Examples / Software



Examples / Software

Example (Your friend, the compiler)

void secret() {
  char* buf = (char*)malloc(1024);
  // put sth. secret into buf
  memset(buf,'\0',1024);
  free(buf);
}

Problem

What if someone gets the same memory?

Examples / Software

Example (Your friend, the compiler)

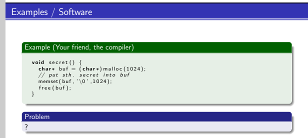
```
void secret() {
  char* buf = (char*)malloc(1024);
  // put sth. secret into buf
  memset(buf,'\0',1024);
  free(buf);
}
```

Problem

?

2023-07-04

DOS - Side-Channels
└Data remanence
 └Software
  └Examples / Software



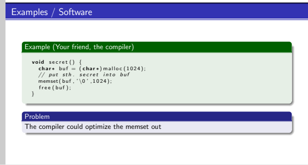
Examples / Software

Example (Your friend, the compiler)

```
void secret() {
  char* buf = (char*)malloc(1024);
  // put sth. secret into buf
  memset(buf,'\0',1024);
  free(buf);
}
```

Problem

The compiler could optimize the memset out



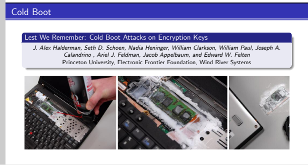
Cold Boot

Lest We Remember: Cold Boot Attacks on Encryption Keys

J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino , Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten
Princeton University, Electronic Frontier Foundation, Wind River Systems

2023-07-04

DOS - Side-Channels
└Data remanence
 └Hardware
  └Cold Boot



Cold Boot

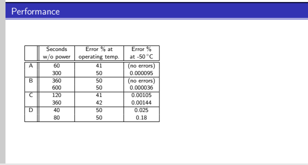
Lest We Remember: Cold Boot Attacks on Encryption Keys

J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino , Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten
Princeton University, Electronic Frontier Foundation, Wind River Systems

Performance

	Seconds w/o power	Error % at operating temp.	Error % at -50 °C
A	60	41	(no errors)
	300	50	0.000095
B	360	50	(no errors)
	600	50	0.000036
C	120	41	0.00105
	360	42	0.00144
D	40	50	0.025
	80	50	0.18

DOS - Side-Channels
└Data remanence
└Hardware
└Performance

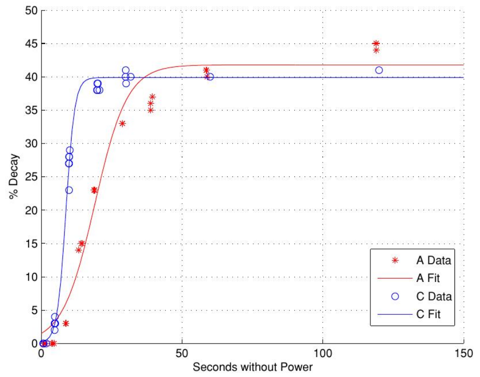


Performance

	Seconds w/o power	Error % at operating temp.	Error % at -50 °C
A	60	41	(no errors)
	300	50	0.000095
B	360	50	(no errors)
	600	50	0.000036
C	120	41	0.00105
	360	42	0.00144
D	40	50	0.025
	80	50	0.18

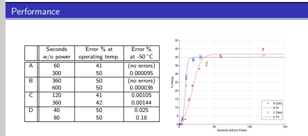
Performance

	Seconds w/o power	Error % at operating temp.	Error % at -50 °C
A	60	41	(no errors)
	300	50	0.000095
B	360	50	(no errors)
	600	50	0.000036
C	120	41	0.00105
	360	42	0.00144
D	40	50	0.025
	80	50	0.18



2023-07-04

DOS - Side-Channels
└Data remanence
 └Hardware
  └Performance



Results

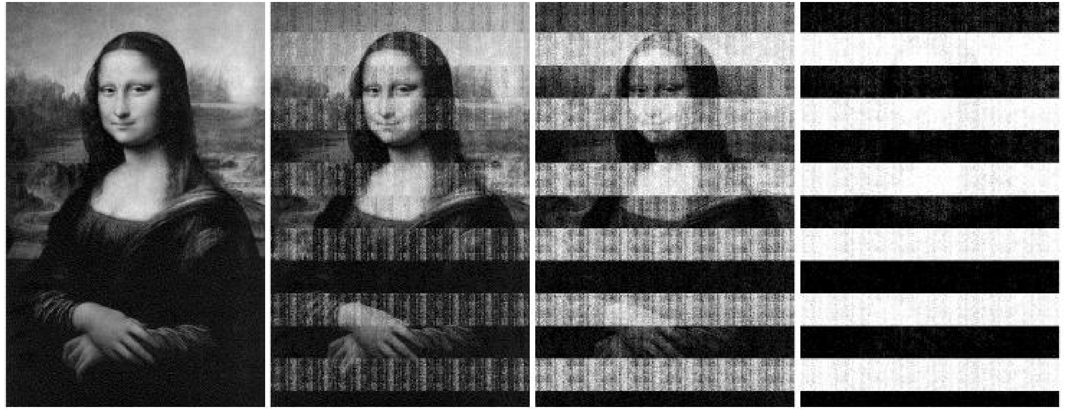
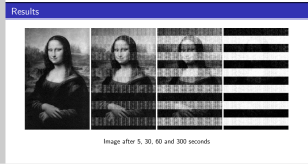


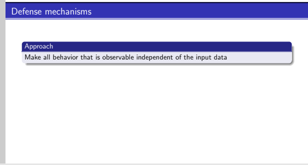
Image after 5, 30, 60 and 300 seconds



Defense mechanisms

Approach

Make all behavior that is observable independent of the input data



Defense mechanisms

Approach

Make all behavior that is observable independent of the input data

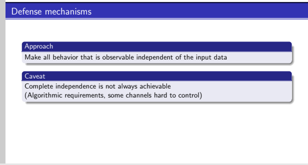
Defense mechanisms

Approach

Make all behavior that is observable independent of the input data

Caveat

Complete independence is not always achievable
(Algorithmic requirements, some channels hard to control)



Defense mechanisms
Approach
Make all behavior that is observable independent of the input data
Caveat
Complete independence is not always achievable
(Algorithmic requirements, some channels hard to control)

Defense mechanisms

Approach

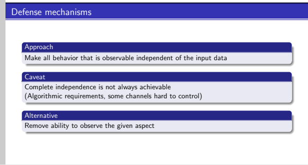
Make all behavior that is observable independent of the input data

Caveat

Complete independence is not always achievable
(Algorithmic requirements, some channels hard to control)

Alternative

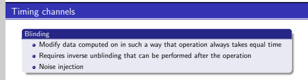
Remove ability to observe the given aspect


Defense mechanisms
Approach
Make all behavior that is observable independent of the input data
Caveat
Complete independence is not always achievable
(Algorithmic requirements, some channels hard to control)
Alternative
Remove ability to observe the given aspect

Timing channels

Blinding

- Modify data computed on in such a way that operation always takes equal time
- Requires inverse unblinding that can be performed after the operation
- Noise injection



Timing channels

Blinding

- Modify data computed on in such a way that operation always takes equal time
- Requires inverse unblinding that can be performed after the operation
- Noise injection

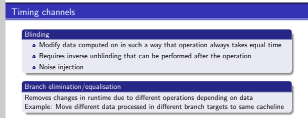
Timing channels

Blinding

- Modify data computed on in such a way that operation always takes equal time
- Requires inverse unblinding that can be performed after the operation
- Noise injection

Branch elimination/equalisation

Removes changes in runtime due to different operations depending on data
Example: Move different data processed in different branch targets to same cacheline


Timing channels
Blinding
- Modify data computed on in such a way that operation always takes equal time
- Requires inverse unblinding that can be performed after the operation
- Noise injection
Branch elimination/equalisation
Removes changes in runtime due to different operations depending on data
Example: Move different data processed in different branch targets to same cacheline

Timing channels

Blinding

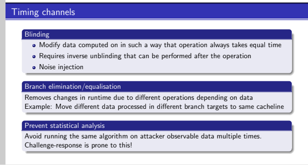
- Modify data computed on in such a way that operation always takes equal time
- Requires inverse unblinding that can be performed after the operation
- Noise injection

Branch elimination/equalisation

Removes changes in runtime due to different operations depending on data
Example: Move different data processed in different branch targets to same cacheline

Prevent statistical analysis

Avoid running the same algorithm on attacker observable data multiple times.
Challenge-response is prone to this!



Timing channels

Blinding
- Modify data computed on in such a way that operation always takes equal time
- Requires inverse unblinding that can be performed after the operation
- Noise injection

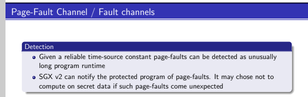
Branch elimination/equalisation
Removes changes in runtime due to different operations depending on data
Example: Move different data processed in different branch targets to same cacheline

Prevent statistical analysis
Avoid running the same algorithm on attacker observable data multiple times.
Challenge-response is prone to this!

Page-Fault Channel / Fault channels

Detection

- Given a reliable time-source constant page-faults can be detected as unusually long program runtime
- SGX v2 can notify the protected program of page-faults. It may chose not to compute on secret data if such page-faults come unexpected


Page-Fault Channel / Fault channels
Detection
Given a reliable time-source constant page-faults can be detected as unusually long program runtime
SGX v2 can notify the protected program of page-faults. It may chose not to compute on secret data if such page-faults come unexpected

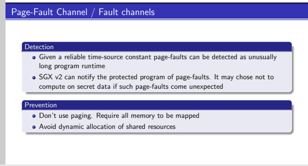
Page-Fault Channel / Fault channels

Detection

- Given a reliable time-source constant page-faults can be detected as unusually long program runtime
- SGX v2 can notify the protected program of page-faults. It may chose not to compute on secret data if such page-faults come unexpected

Prevention

- Don't use paging. Require all memory to be mapped
- Avoid dynamic allocation of shared resources



Page-Fault Channel / Fault channels

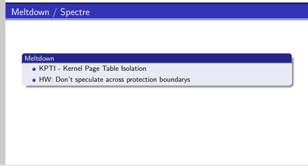
Detection
- Given a reliable time-source constant page-faults can be detected as unusually long program runtime
- SGX v2 can notify the protected program of page-faults. It may chose not to compute on secret data if such page-faults come unexpected

Prevention
- Don't use paging. Require all memory to be mapped
- Avoid dynamic allocation of shared resources

Meltdown / Spectre

Meltdown

- KPTI - Kernel Page Table Isolation
- HW: Don't speculate across protection boundarys



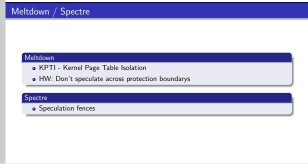
Meltdown / Spectre

Meltdown

- KPTI - Kernel Page Table Isolation
- HW: Don't speculate across protection boundarys

Spectre

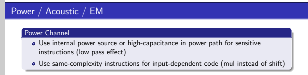
- Speculation fences


Meltdown / Spectre
Meltdown
- KPTI - Kernel Page Table Isolation
- HW: Don't speculate across protection boundarys
Spectre
- Speculation fences

Power / Acoustic / EM

Power Channel

- Use internal power source or high-capacitance in power path for sensitive instructions (low pass effect)
- Use same-complexity instructions for input-dependent code (mul instead of shift)


Power / Acoustic / EM

Power Channel

- Use internal power source or high-capacitance in power path for sensitive instructions (low pass effect)
- Use same-complexity instructions for input-dependent code (mul instead of shift)

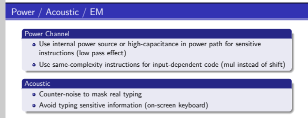
Power / Acoustic / EM

Power Channel

- Use internal power source or high-capacitance in power path for sensitive instructions (low pass effect)
- Use same-complexity instructions for input-dependent code (mul instead of shift)

Acoustic

- Counter-noise to mask real typing
- Avoid typing sensitive information (on-screen keyboard)


Power / Acoustic / EM
Power Channel
- Use internal power source or high-capacitance in power path for sensitive instructions (low pass effect)
- Use same-complexity instructions for input-dependent code (mul instead of shift)
Acoustic
- Counter-noise to mask real typing
- Avoid typing sensitive information (on-screen keyboard)

Power / Acoustic / EM

Power Channel

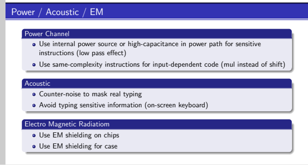
- Use internal power source or high-capacitance in power path for sensitive instructions (low pass effect)
- Use same-complexity instructions for input-dependent code (mul instead of shift)

Acoustic

- Counter-noise to mask real typing
- Avoid typing sensitive information (on-screen keyboard)

Electro Magnetic Radiatiom

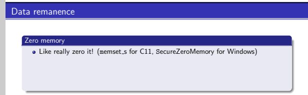
- Use EM shielding on chips
- Use EM shielding for case


Power / Acoustic / EM
Power Channel
- Use internal power source or high-capacitance in power path for sensitive instructions (low pass effect)
- Use same-complexity instructions for input-dependent code (mul instead of shift)
Acoustic
- Counter-noise to mask real typing
- Avoid typing sensitive information (on-screen keyboard)
Electro Magnetic Radiatiom
- Use EM shielding on chips
- Use EM shielding for case

Data remanence

Zero memory

- Like really zero it! (`memset_s` for C11, `SecureZeroMemory` for Windows)


Data remanence

Zero memory

- Like really zero it! (memset_s for C11, SecureZeroMemory for Windows)

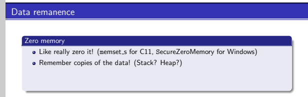
Data remanence

Zero memory

- Like really zero it! (`memset_s` for C11, `SecureZeroMemory` for Windows)
- Remember copies of the data! (Stack? Heap?)

2023-07-04

DOS - Side-Channels
└Defense
└Data remanence


Data remanence

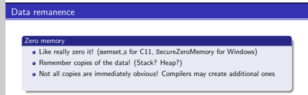
Zero memory

- Like really zero it! (memset_s for C11, SecureZeroMemory for Windows)
- Remember copies of the data! (Stack? Heap?)

Data remanence

Zero memory

- Like really zero it! (`memset_s` for C11, `SecureZeroMemory` for Windows)
- Remember copies of the data! (Stack? Heap?)
- Not all copies are immediately obvious! Compilers may create additional ones


Data remanence

Zero memory

- Like really zero it! (memset_s for C11, SecureZeroMemory for Windows)
- Remember copies of the data! (Stack? Heap?)
- Not all copies are immediately obvious! Compilers may create additional ones

Data remanence

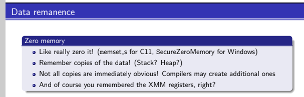
Zero memory

- Like really zero it! (`memset_s` for C11, `SecureZeroMemory` for Windows)
- Remember copies of the data! (Stack? Heap?)
- Not all copies are immediately obvious! Compilers may create additional ones
- And of course you remembered the XMM registers, right?

2023-07-04

DOS - Side-Channels
└Defense

└Data remanence


Data remanence

Zero memory

- Like really zero it! (memset_s for C11, SecureZeroMemory for Windows)
- Remember copies of the data! (Stack? Heap?)
- Not all copies are immediately obvious! Compilers may create additional ones
- And of course you remembered the XMM registers, right?

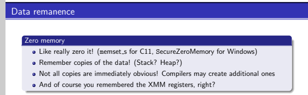
Data remanence

Zero memory

- Like really zero it! (`memset_s` for C11, `SecureZeroMemory` for Windows)
- Remember copies of the data! (Stack? Heap?)
- Not all copies are immediately obvious! Compilers may create additional ones
- And of course you remembered the XMM registers, right?

2023-07-04

DOS - Side-Channels
└Defense
└Data remanence


Data remanence

Zero memory
- Like really zero it! (memset_s for C11, SecureZeroMemory for Windows)
- Remember copies of the data! (Stack? Heap?)
- Not all copies are immediately obvious! Compilers may create additional ones
- And of course you remembered the XMM registers, right?

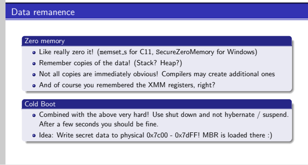
Data remanence

Zero memory

- Like really zero it! (`memset_s` for C11, `SecureZeroMemory` for Windows)
- Remember copies of the data! (Stack? Heap?)
- Not all copies are immediately obvious! Compilers may create additional ones
- And of course you remembered the XMM registers, right?

Cold Boot

- Combined with the above very hard! Use shut down and not hybernate / suspend. After a few seconds you should be fine.
- Idea: Write secret data to physical 0x7c00 - 0x7dFF! MBR is loaded there :)



Data remanence

Zero memory
- Like really zero it! (memset_s for C11, SecureZeroMemory for Windows)
- Remember copies of the data! (Stack? Heap?)
- Not all copies are immediately obvious! Compilers may create additional ones
- And of course you remembered the XMM registers, right?

Cold Boot
- Combined with the above very hard! Use shut down and not hybernate / suspend. After a few seconds you should be fine.
- Idea: Write secret data to physical 0x7c00 - 0x7dFF! MBR is loaded there :)

Summary

Sidechannels

… are unintended information sources for extracting secret data

2023-07-04

DOS - Side-Channels
└Conclusion
  └Summary
    └Summary

Summary

Sidechannels

… are unintended information sources for extracting secret data

Summary

Sidechannels

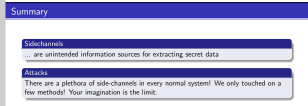
... are unintended information sources for extracting secret data

Attacks

There are a plethora of side-channels in every normal system! We only touched on a few methods! Your imagination is the limit.

2023-07-04

DOS - Side-Channels
└Conclusion
 └Summary
  └Summary


Summary

Sidechannels
... are unintended information sources for extracting secret data

Attacks
There are a plethora of side-channels in every normal system! We only touched on a few methods! Your imagination is the limit.

Summary

Sidechannels

… are unintended information sources for extracting secret data

Attacks

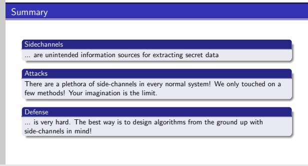
There are a plethora of side-channels in every normal system! We only touched on a few methods! Your imagination is the limit.

Defense

… is very hard. The best way is to design algorithms from the ground up with side-channels in mind!

2023-07-04

DOS - Side-Channels
└Conclusion
 └Summary
  └Summary



Summary

Sidechannels
… are unintended information sources for extracting secret data

Attacks
There are a plethora of side-channels in every normal system! We only touched on a few methods! Your imagination is the limit.

Defense
… is very hard. The best way is to design algorithms from the ground up with side-channels in mind!

Overview

- http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-3/physec/papers/physecpaper19.pdf

Cache Side-Channels

- https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-yarom.pdf

Page-fault Channel

- http://www.ieee-security.org/TC/SP2015/papers-archived/6949a640.pdf
- https://www.usenix.org/system/files/conference/atc17/atc17-hahnel.pdf

Microarchitectural Channels

- https://meltdownattack.com/meltdown.pdf
- https://spectreattack.com/spectre.pdf

Acoustic Channels

- http://people.eecs.berkeley.edu/ tygar/papers/Keyboard_Acoustic_Emanations_Revisited/ccs.pdf

Overview
- http://csrc.nist.gov/groups/STM/cmvp/documents/fips140-3/physec/papers/physecpaper19.pdf
Cache Side-Channels
- https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-yarom.pdf
Page-fault Channel
- http://www.ieee-security.org/TC/SP2015/papers-archived/6949a640.pdf
- https://www.usenix.org/system/files/conference/atc17/atc17-hahnel.pdf
Microarchitectural Channels
- https://meltdownattack.com/meltdown.pdf
- https://spectreattack.com/spectre.pdf
Acoustic Channels
- http://people.eecs.berkeley.edu/ tygar/papers/Keyboard_Acoustic_Emanations_Revisited/ccs.pdf

Cold Boot

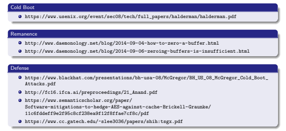
- https://www.usenix.org/event/sec08/tech/full_papers/halderman/halderman.pdf

Remanence

- http://www.daemonology.net/blog/2014-09-04-how-to-zero-a-buffer.html
- http://www.daemonology.net/blog/2014-09-06-zeroing-buffers-is-insufficient.html

Defense

- https://www.blackhat.com/presentations/bh-usa-08/McGregor/BH_US_08_McGregor_Cold_Boot_Attacks.pdf
- http://fc16.ifca.ai/preproceedings/21_Anand.pdf
- https://www.semanticscholar.org/paper/Software-mitigations-to-hedge-AES-against-cache-Brickell-Graunke/11c6fddeff9e2f95c8cf238ea9f12f8ffae7cf8c/pdf
- https://www.cc.gatech.edu/~slee3036/papers/shih:tsgx.pdf



Cold Boot
- https://www.usenix.org/event/sec08/tech/full_papers/halderman/halderman.pdf

Remanence
- http://www.daemonology.net/blog/2014-09-04-how-to-zero-a-buffer.html
- http://www.daemonology.net/blog/2014-09-06-zeroing-buffers-is-insufficient.html

Defense
- https://www.blackhat.com/presentations/bh-usa-08/McGregor/BH_US_08_McGregor_Cold_Boot_Attacks.pdf
- http://fc16.ifca.ai/preproceedings/21_Anand.pdf
- https://www.semanticscholar.org/paper/Software-mitigations-to-hedge-AES-against-cache-Brickell-Graunke/11c6fddeff9e2f95c8cf238ea9f12f8ffae7cf8c/pdf
- https://www.cc.gatech.edu/~slee3036/papers/shih:tsgx.pdf