High Resolution Side Channels for Untrusted Operating Systems

Marcus Hähnel [1] Marcus Peinado [2] Weidong Cui [2]

[1]TU Dresden

[2]Microsoft Research

2017-07-13

Reasons to distrust the OS

Reasons to distrust the OS

Reasons to distrust the OS

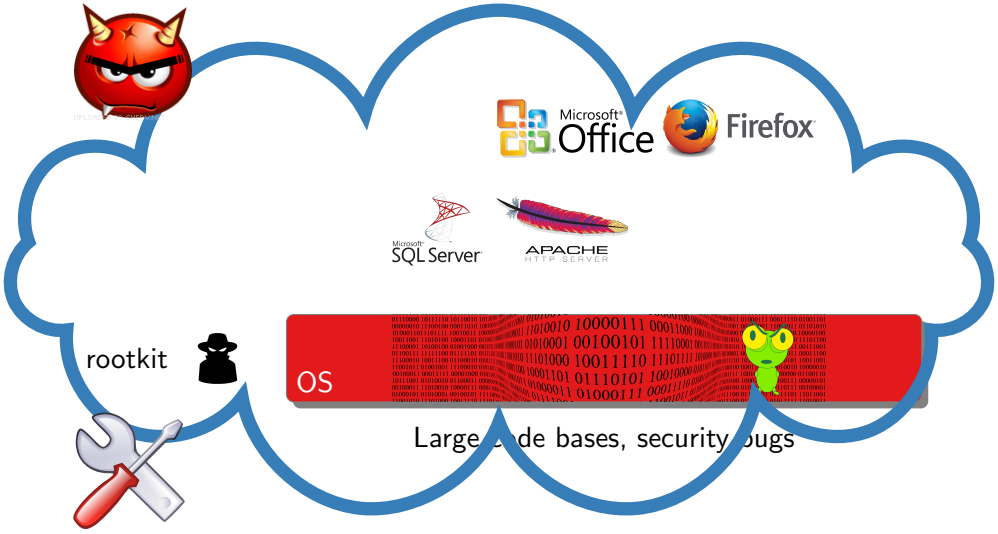
Large code bases, security bugs

Reasons to distrust the OS

rootkit 

Large code bases, security bugs

Reasons to distrust the OS



Shielding Systems

Shielding Systems

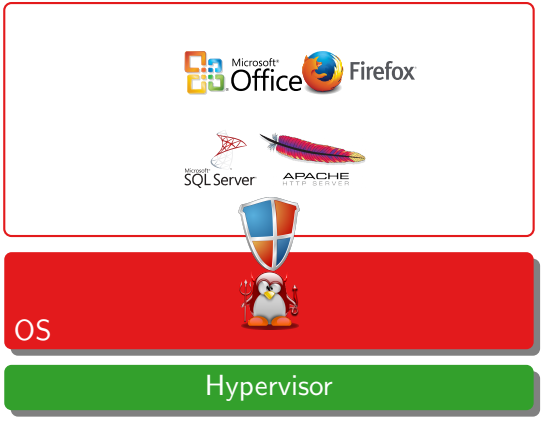
Removing the OS from the trusted computing base

Shielding Systems

Removing the OS from the trusted computing base

Hypervisor-based

- Overshadow [ASPLOS'08]
- InkTag [ASPLOS'13]



Shielding Systems

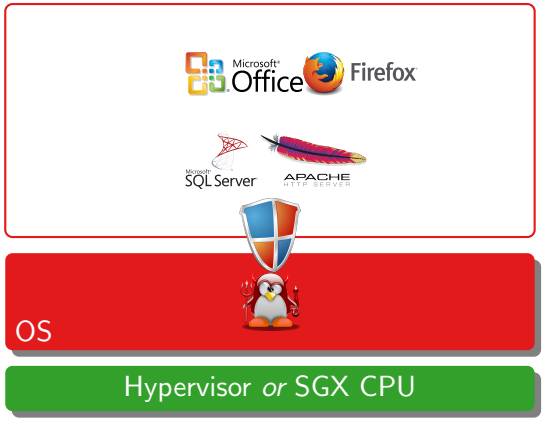
Removing the OS from the trusted computing base

Hypervisor-based

- Overshadow [ASPLOS'08]
- InkTag [ASPLOS'13]

Intel SGX-based

- Haven [OSDI'14]
- VC3 [Oakland'15]
- SCONE [OSDI'16]
- Glamdring [ATC'17]



Shielding Systems

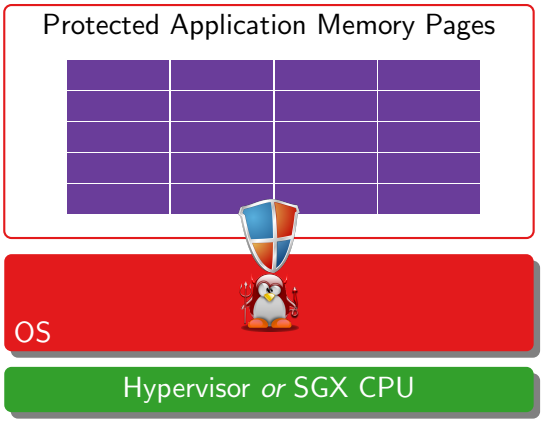
Removing the OS from the trusted computing base

Hypervisor-based

- Overshadow [ASPLOS'08]
- InkTag [ASPLOS'13]

Intel SGX-based

- Haven [OSDI'14]
- VC3 [Oakland'15]
- SCONE [OSDI'16]
- Glamdring [ATC'17]



Attack position

But how well do these solutions protect the application?

Controlled Channels [1]

OS still manages shielded applications

- Control over page tables
- ... and thus over page faults 😇

[1] Xu, Yuanzhong, Weidong Cui, and Marcus Peinado. "Controlled-channel attacks: Deterministic side channels for untrusted operating systems.", Oakland 2015

Controlled Channels [1]

OS still manages shielded applications

- Control over page tables
- ... and thus over page faults 😇

Data dependent control flow

```
// @ Page 1
void processData(bool secret) {
    if (secret) {
        secretData(); // @ Page 2
    } else {
        publicData(); // @ Page 3
    }
}
```

[1] Xu, Yuanzhong, Weidong Cui, and Marcus Peinado. "Controlled-channel attacks: Deterministic side channels for untrusted operating systems.", Oakland 2015

Controlled Channels [1]

OS still manages shielded applications

- Control over page tables
- ... and thus over page faults 😇

Data dependent control flow

```
// @ Page 1
void processData(bool secret) {
    if (secret) {
        secretData(); // @ Page 2
    } else {
        publicData(); // @ Page 3
    }
}
```

Page faults serve as de facto *breakpoints* and reveal memory access patterns

[1] Xu, Yuanzhong, Weidong Cui, and Marcus Peinado. "Controlled-channel attacks: Deterministic side channels for untrusted operating systems.", Oakland 2015

Controlled Channels [1]

OS still manages shielded applications

- Control over page tables
- ... and thus over page faults 😇

Retrieved

- outlines of images
- text from font rendering
- text from spell checking

Data dependent control flow

```
// @ Page 1
void processData(bool secret) {
    if (secret) {
        secretData(); // @ Page 2
    } else {
        publicData(); // @ Page 3
    }
}
```

Page faults serve as de facto *breakpoints* and reveal memory access patterns

[1] Xu, Yuanzhong, Weidong Cui, and Marcus Peinado. "Controlled-channel attacks: Deterministic side channels for untrusted operating systems.", Oakland 2015

Contributions

Page-Fault Channel Limitations

- limited to page granular memory observation

Contributions

Page-Fault Channel Limitations

- limited to page granular memory observation
- requires *page toggling*

Contributions

Page-Fault Channel Limitations

- limited to page granular memory observation
- requires *page toggling*
- is *only* means to set breakpoint (may be detectable)

Contributions

Page-Fault Channel Limitations

- limited to page granular memory observation
- requires *page toggling*
- is *only* means to set breakpoint (may be detectable)

Table 2-4. Bit Vector Layout of MISCSELECT Field of Extended Information

Field	Bit Position	Description
EXINFO	0	Report page fault and general protection exception info inside an enclave
Reserved	31:1	Reserved (0).

Contributions

Page-Fault Channel Limitations

- limited to page granular memory observation
  - ⇒ Increase spatial resolution
- requires *page toggling*
  - ⇒ Improve temporal resolution
- is *only* means to set breakpoint (may be detectable)

Table 2-4. Bit Vector Layout of MISCSELECT Field of Extended Information

Field	Bit Position	Description
EXINFO	0	Report page fault and general protection exception info inside an enclave
Reserved	31:1	Reserved (0).

Contributions

Page-Fault Channel Limitations

- limited to page granular memory observation
  - ⇒ Increase spatial resolution
- requires *page toggling*
  - ⇒ Improve temporal resolution
- is *only* means to set breakpoint (may be detectable)

Show more code than previously thought is vulnerable

Table 2-4. Bit Vector Layout of MISCSELECT Field of Extended Information

Field	Bit Position	Description
EXINFO	0	Report page fault and general protection exception info inside an enclave
Reserved	31:1	Reserved (0).

Contributions

Page-Fault Channel Limitations

- limited to page granular memory observation
  - ⇒ Increase spatial resolution
- requires *page toggling*
  - ⇒ Improve temporal resolution
- is *only* means to set breakpoint (may be detectable)
  - ⇒ Other ways to step through the application

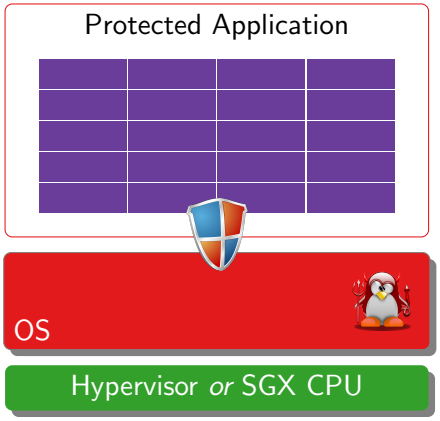
Show more code than previously thought is vulnerable

Table 2-4. Bit Vector Layout of MISCSELECT Field of Extended Information

Field	Bit Position	Description
EXINFO	0	Report page fault and general protection exception info inside an enclave
Reserved	31:1	Reserved (0).

System Model

Assumptions

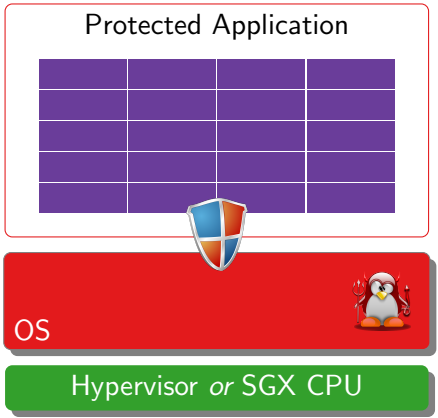


System Model

Assumptions

Working shielding system

... protects integrity and security of applications' memory against direct access



System Model

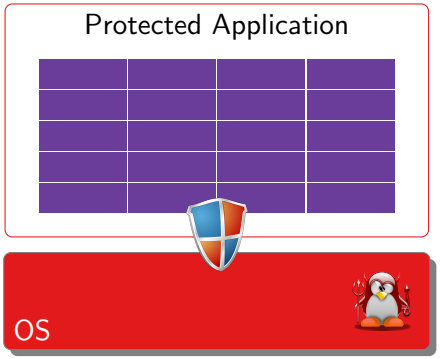
Assumptions

Working shielding system

... protects integrity and security of applications' memory against direct access

Commodity OS

... is still responsible for:



System Model

Assumptions

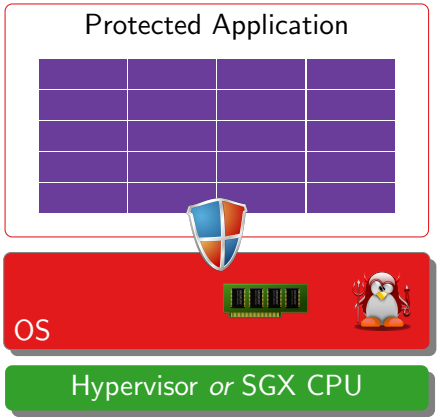
Working shielding system

... protects integrity and security of applications' memory against direct access

Commodity OS

... is still responsible for:

- Memory management



System Model

Assumptions

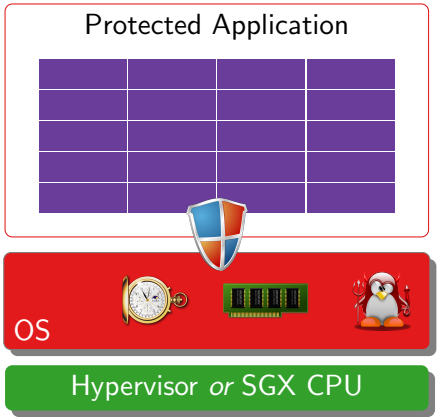
Working shielding system

... protects integrity and security of applications' memory against direct access

Commodity OS

... is still responsible for:

- Memory management
- Scheduling



System Model

Assumptions

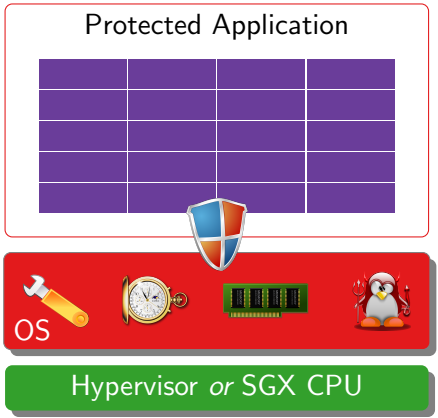
Working shielding system

... protects integrity and security of applications' memory against direct access

Commodity OS

... is still responsible for:

- Memory management
- Scheduling
- Hardware Configuration



New Attack Tools

Timer-based Attacks

Scheduling

The OS has control over scheduling ... and thus over timers 

Timer-based Attacks

Scheduling

The OS has control over scheduling ... and thus over timers 😇

Challenges

- 25 MHz LAPIC Timer vs. 4 GHz CPU clock

Timer-based Attacks

Scheduling

The OS has control over scheduling ... and thus over timers 😇

Challenges

- 25 MHz LAPIC Timer vs. 4 GHz CPU clock
- No page fault address

Timer-based Attacks

Scheduling

The OS has control over scheduling ... and thus over timers 😇

Challenges

- 25 MHz LAPIC Timer vs. 4 GHz CPU clock
- No page fault address

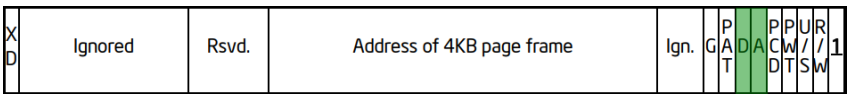


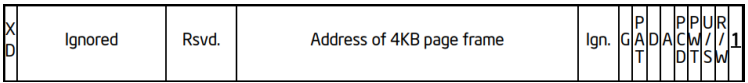
Figure: Accessed & Dirty bits in PTE

Example

```
size_t strlen(const char* str) {
    size_t len = 0;
    while (*str != '\0') {
        str++;
        len++;
    }
    return len;
}

const char* s = "The";
int l = strlen(s);
```

*str	Attacker count
'T'	0



Example

```
size_t strlen(const char* str) {
    size_t len = 0;
    while (*str != '\0') {
        str++;
        len++;
    }
    return len;
}

const char* s = "The";
int l = strlen(s);
```

*str	Attacker count
'T'	0

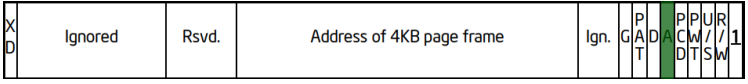
XD	Ignored	Rsvd.	Address of 4KB page frame	Ign.	G	PAT	D	A	PCD	PWT	U/S	R/W	1

Example

```
size_t strlen(const char* str) {
    size_t len = 0;
    while (*str != '\0') {
        str++;
        len++;
    }
    return len;
}

const char* s = "The";
int l = strlen(s);
```

*str	Attacker count
'T'	0

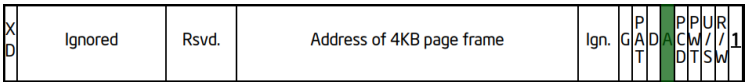


Example

```
size_t strlen(const char* str) {
    size_t len = 0;
    while (*str != '\0') {
        str++;
        len++;
    }
    return len;
}

const char* s = "The";
int l = strlen(s);
```

*str	Attacker count
'T'	0

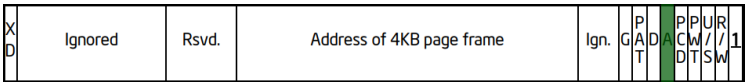


Example

```
size_t strlen(const char* str) {
    size_t len = 0;
    while (*str != '\0') {
        str++;
        len++;
    }
    return len;
}

const char* s = "The";
int l = strlen(s);
```

*str	Attacker count
'T'	1

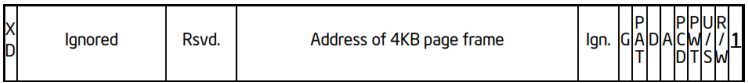


Example

```
size_t strlen(const char* str) {
    size_t len = 0;
    while (*str != '\0') {
        str++;
        len++;
    }
    return len;
}

const char* s = "The";
int l = strlen(s);
```

*str	Attacker count
'T'	1



Example

```
size_t strlen(const char* str) {
    size_t len = 0;
    while (*str != '\0') {
        str++;
        len++;
    }
    return len;
}

const char* s = "The";
int l = strlen(s);
```

*str	Attacker count
'T'	1
'h'	1

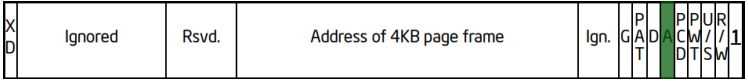
XD	Ignored	Rsvd.	Address of 4KB page frame	Ign.	G	PAT	D	A	PCD	PWT	U/S	R/W	1

Example

```
size_t strlen(const char* str) {
    size_t len = 0;
    while (*str != '\0') {
        str++;
        len++;
    }
    return len;
}

const char* s = "The";
int l = strlen(s);
```

*str	Attacker count
'T'	1
'h'	2
'e'	3
'\0'	4



Results

STRLEN function

- 99.98 % of string lengths detected correctly
- Can *effectively* single-step through the application
- Works where Page-Fault Channel fails
- Can replace page-fault based break points
- Requires fine-tuning for correct timing

Prime & Probe

Traditional Cache Side-Channel

- Unprivileged attacker and victim on same machine share cache
- Attacker can indirectly observe victims memory access

Prime & Probe

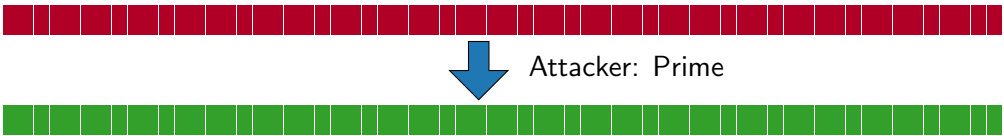
Traditional Cache Side-Channel

- Unprivileged attacker and victim on same machine share cache
- Attacker can indirectly observe victims memory access

Prime & Probe

Traditional Cache Side-Channel

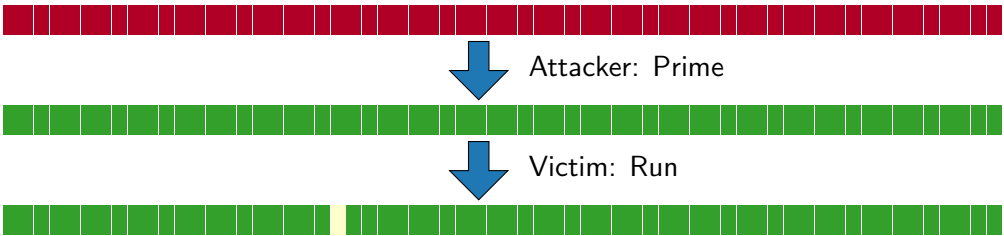
- Unprivileged attacker and victim on same machine share cache
- Attacker can indirectly observe victims memory access



Prime & Probe

Traditional Cache Side-Channel

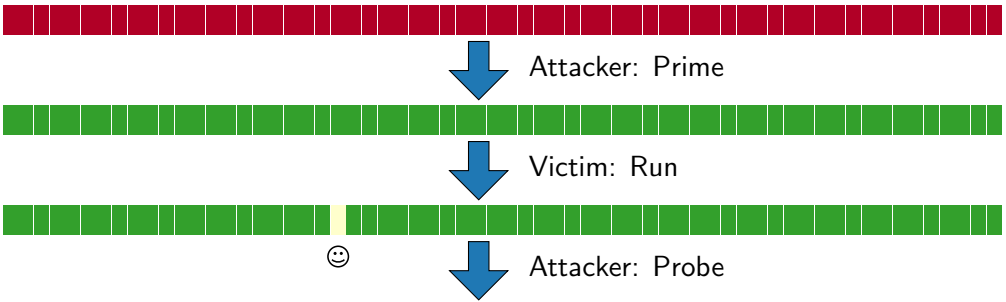
- Unprivileged attacker and victim on same machine share cache
- Attacker can indirectly observe victims memory access



Prime & Probe

Traditional Cache Side-Channel

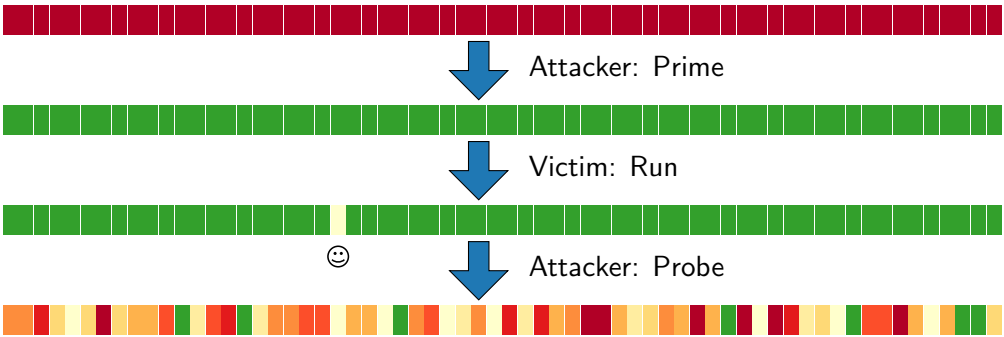
- Unprivileged attacker and victim on same machine share cache
- Attacker can indirectly observe victims memory access



Prime & Probe

Traditional Cache Side-Channel

- Unprivileged attacker and victim on same machine share cache
- Attacker can indirectly observe victims memory access



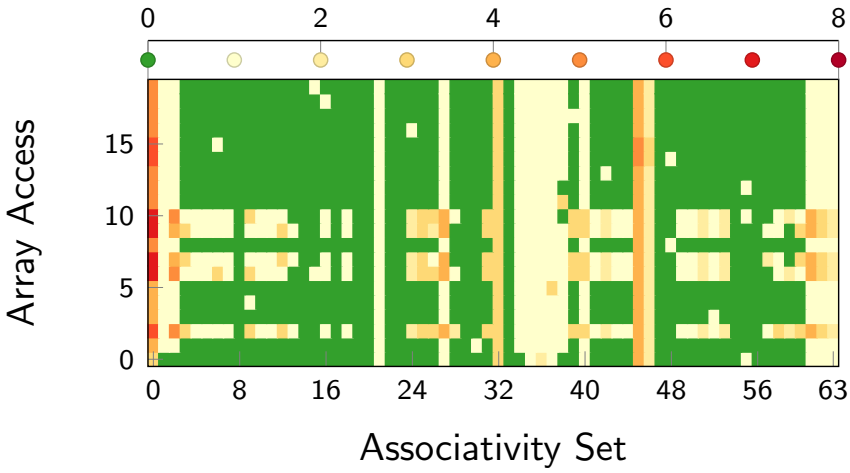
Prime & Probe

But we are not an unprivileged attacker, but the OS

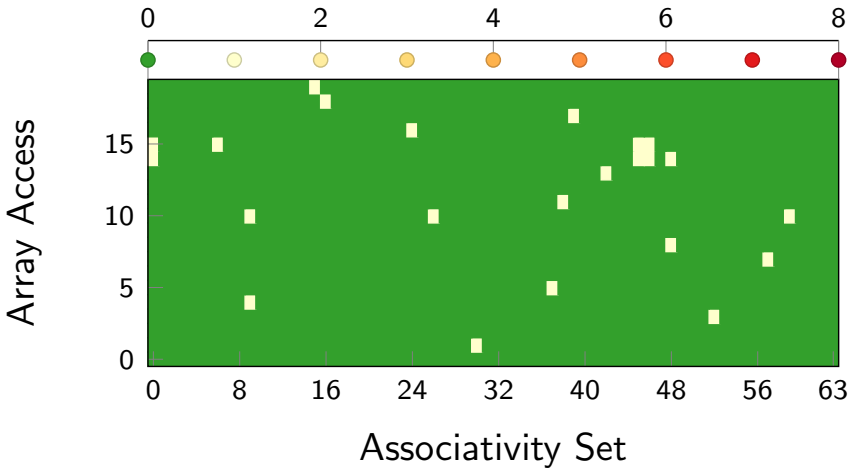
Noise reduction by

- Targeted Breakpoints
- Preventing other applications from being scheduled
- Turn off prefetching

Results


0
2
4
6
8
Array Access
0
5
10
15
0
8
16
24
32
40
48
56
63
Associativity Set

Results

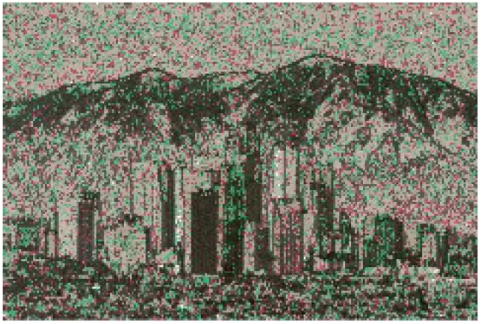

0
2
4
6
8
Array Access
15
10
5
0
0
8
16
24
32
40
48
56
63
Associativity Set

Evaluation

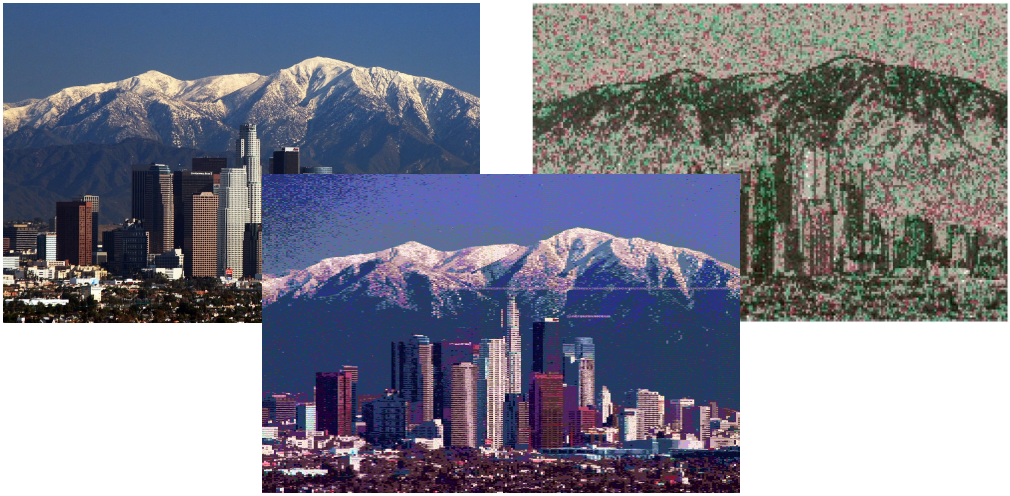
Victims

- libjpeg: image decoding
- VC3: map-reduce framework for SGX

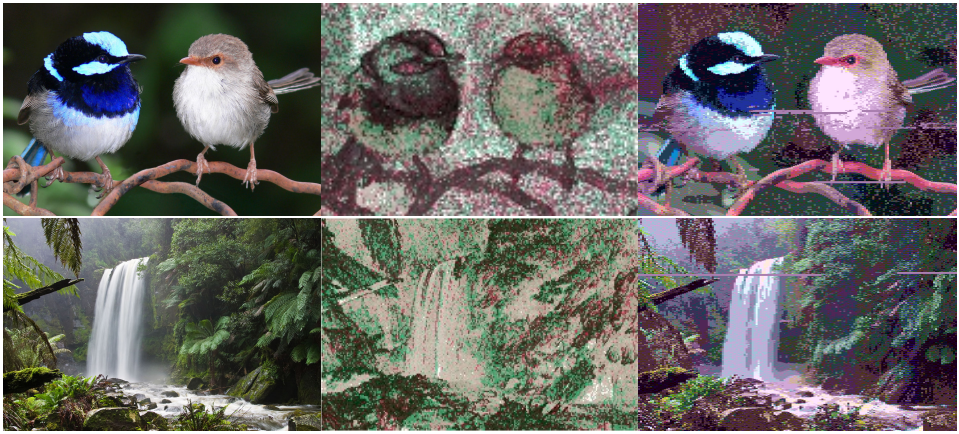
libjpeg: High resolution image extraction



libjpeg: High resolution image extraction



libjpeg: High resolution image extraction



VC3

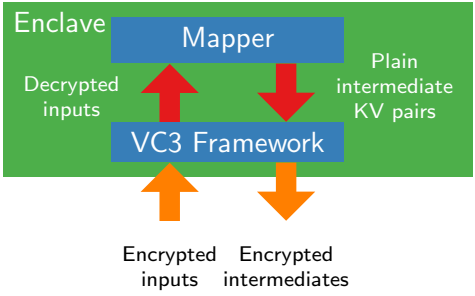
Why is attacking VC3 interesting

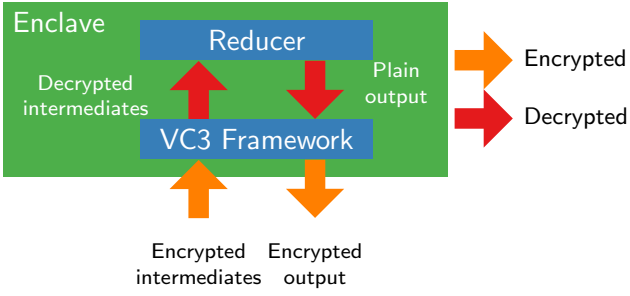
- First/only realistic shielding system for Hadoop
- Protects mapper and reducer applications and their data from the OS/cloud
- Uses SGX (Enclaves)

VC3

Why is attacking VC3 interesting

- First/only realistic shielding system for Hadoop
- Protects mapper and reducer applications and their data from the OS/cloud
- Uses SGX (Enclaves)





Attack Overview

Why is attacking VC3 *hard*

- Only attack framework; not user's secret mappers and reducers
- Framework is small (only 13 code pages)
- Framework does not know application semantics

Attack Overview

Why is attacking VC3 *hard*

- Only attack framework; not user's secret mappers and reducers
- Framework is small (only 13 code pages)
- Framework does not know application semantics

Can this leak information?

Attack Overview

Why is attacking VC3 *hard*

- Only attack framework; not user's secret mappers and reducers
- Framework is small (only 13 code pages)
- Framework does not know application semantics

Can this leak information?

Map/Reduce spec

"The MapReduce library *groups together* all intermediate values associated with the same intermediate key *I* and passes them to the *Reduce* function" [2]

VC3 implements grouping using a hash table 😇

[2] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51.1 (2008): 107-103. [Page 2]

Attack: Information Gathering Phase

Victim

Concrete attack here: *WordCount* (or similar: e.g., *inverted index*)

Attack: Information Gathering Phase

Victim

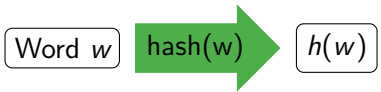
Concrete attack here: *WordCount* (or similar: e.g., *inverted index*)

Word w

Attack: Information Gathering Phase

Victim

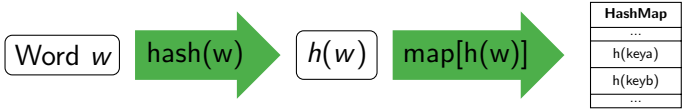
Concrete attack here: *WordCount* (or similar: e.g., *inverted index*)



Attack: Information Gathering Phase

Victim

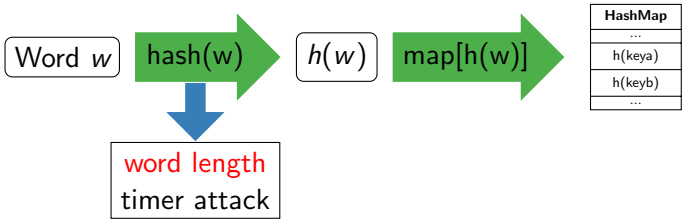
Concrete attack here: *WordCount* (or similar: e.g., *inverted index*)



Attack: Information Gathering Phase

Victim

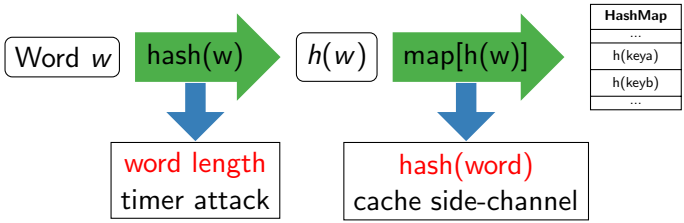
Concrete attack here: *WordCount* (or similar: e.g., *inverted index*)



Attack: Information Gathering Phase

Victim

Concrete attack here: *WordCount* (or similar: e.g., *inverted index*)



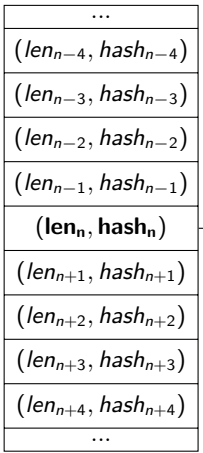
Attack: Text Recovery Phase

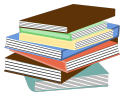
Using the gathered information to re-construct the document

...
$(len_{n-4}, hash_{n-4})$
$(len_{n-3}, hash_{n-3})$
$(len_{n-2}, hash_{n-2})$
$(len_{n-1}, hash_{n-1})$
$(\mathbf{len_n}, \mathbf{hash_n})$
$(len_{n+1}, hash_{n+1})$
$(len_{n+2}, hash_{n+2})$
$(len_{n+3}, hash_{n+3})$
$(len_{n+4}, hash_{n+4})$
...

Attack: Text Recovery Phase

Using the gathered information to re-construct the document



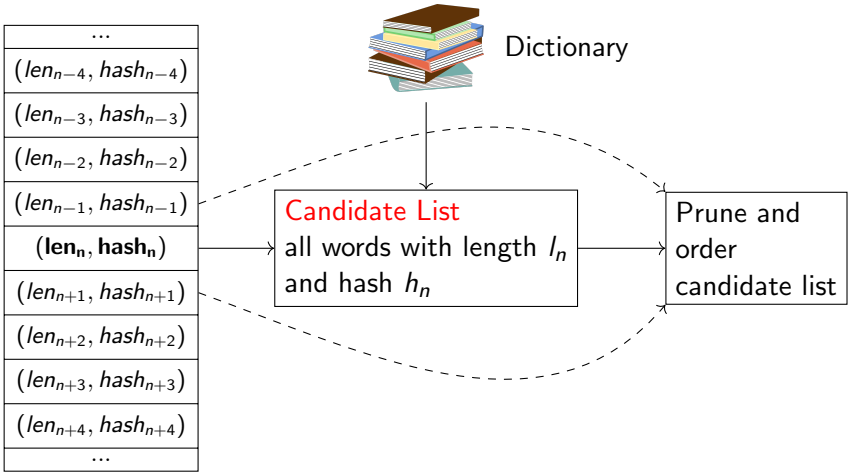


Dictionary

Candidate List
all words with length l_n
and hash h_n

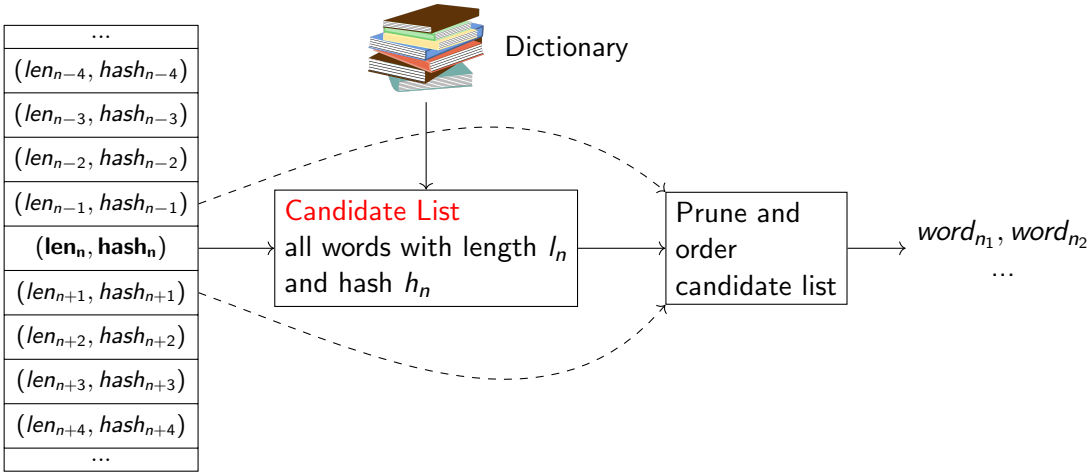
Attack: Text Recovery Phase

Using the gathered information to re-construct the document



Attack: Text Recovery Phase

Using the gathered information to re-construct the document



Oz text recovered

THE WONDERFUL WIZARD OF OZ

The Cyclone

Dorothy lived in the midst of the great kansas prairies with Uncle Henry who was a farmer and Aunt Em who was the Their house was small for the lumber to build it had to be carried by wagon many There were four walls a floor and a roof which made one and this room contained a rusty looking cookstove a cupboard for the dishes a table three or four chairs and the Uncle Henry and Aunt Em had a big bed in one corner and Dorothy a little bed in another There was no garret at all and no a small hole dug in the ground called a cyclone cellar where the family could go in case one of those great whirlwinds arose mighty enough to crush any building in its It was reached by a trap door in the middle of the floor from which a ladder led down into the small dark

When Dorothy stood in the doorway and looked around she could see nothing but the great gray prairie on every Not a tree nor a house broke the broad sweep of flat country that reached to the edge of the sky In all The sun had burned the plowed land into a gray mass with little cracks running through Even the grass was not green for the sun had burned the tops of the long blades until they were the same gray color to be seen Once the house had been painted but the sun blistered the paint and the rains washed it away and now the house was as dull and gray as everything

When Aunt Em came there to live she was a young pretty The sun and wind had changed her They had taken the sparkle from her eyes and left them a sober they had taken the red from her cheeks and lips and they were gray She was thin and never smiled When Dorothy who was an orphan first came to her Aunt Em had been so startled by the laughter that she would scream and press her hand upon her heart whenever merry voice reached her and she still looked at the little girl with wonder that she could find anything to laugh

Conclusion

Enhanced Side-Channels

- memory access detection at higher *spatial* resolution (64 byte vs. 4kB granularity)
- fine-granular *breakpoints* through timers
- low-noise cache side-channel with single execution

Conclusion

Enhanced Side-Channels

- memory access detection at higher *spatial* resolution (64 byte vs. 4kB granularity)
- fine-granular *breakpoints* through timers
- low-noise cache side-channel with single execution

Results

- High resolution image extraction from libjpeg
- Document extraction from map/reduce

Conclusion

Enhanced Side-Channels

- memory access detection at higher *spatial* resolution (64 byte vs. 4kB granularity)
- fine-granular *breakpoints* through timers
- low-noise cache side-channel with single execution

Results

- High resolution image extraction from libjpeg
- Document extraction from map/reduce

Mitigations

Are increasingly important

- T-SGX, Intel Taint Analysis Tool, Trusted Schedulers