TU Graz
SCIENCE
PASSION
TECHNOLOGY

KernelSnitch

Side-Channel Attacks on Kernel Data Structures

Lukas Maar Jonas Juffinger Thomas Steinbauer Daniel Gruss Stefan Mangard

Graz University of Technology, Austria

NDSS 2025

› isec.tugraz.at

About

Novel OS side channel **KernelSnitch**

About

Novel OS side channel **KernelSnitch**

- Timing side channel:
  - Different access timings of kernel data structures

About

Novel OS side channel **KernelSnitch**

- Timing side channel:
  - Different access timings of kernel data structures
- Amplification:
  - Make timing difference exploitable from user space

About

Novel OS side channel **KernelSnitch**

- Timing side channel:
  - Different access timings of kernel data structures
- Amplification:
  - Make timing difference exploitable from user space
- Attacks:
  - Covert channel with up to 580 $\mathrm{kbit\,s^{-1}}$
  - Website fingerprinting with F_1 of 89.3 %
  - Kernel heap pointer leak in under 65 s

About

Novel OS side channel **KernelSnitch**

- Timing side channel:
  - Different access timings of kernel data structures
- Amplification:
  - Make timing difference exploitable from user space
- Attacks:
  - Covert channel with up to 580 $\mathrm{kbit\,s^{-1}}$
  - Website fingerprinting with F_1 of 89.3 %
  - Kernel heap pointer leak in under 65 s
- Opensource our artifacts on:
  - https://github.com/IAIK/KernelSnitch

Motivation & Background

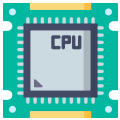
Motivation

Hardware

Application

Motivation

Hardware



Application

Motivation

Hardware

Application

Motivation

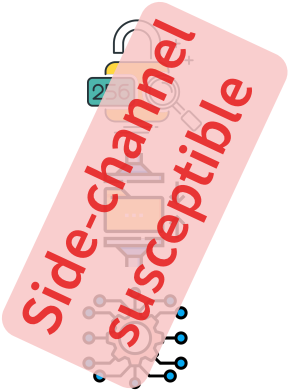
Hardware

Application

Motivation

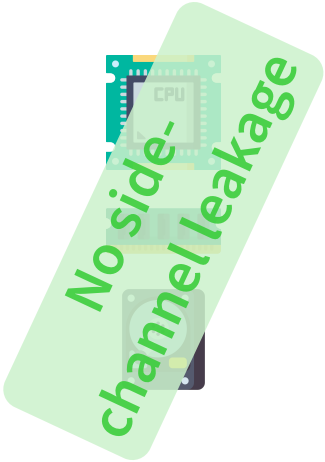
Hardware

Application

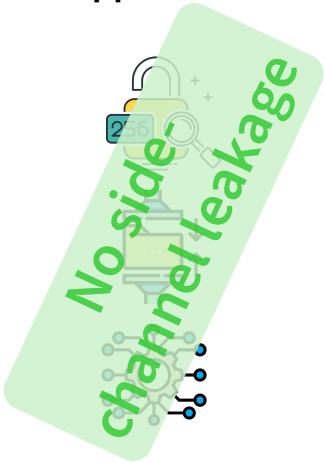


Motivation

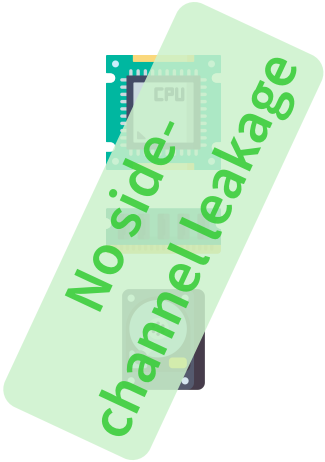
Hardware



Application

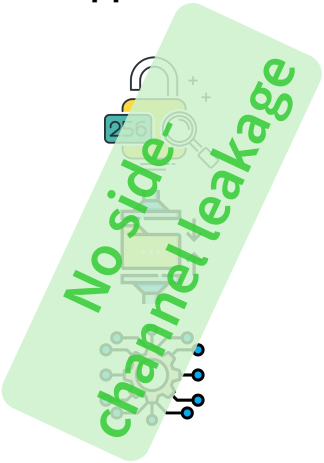


Hardware



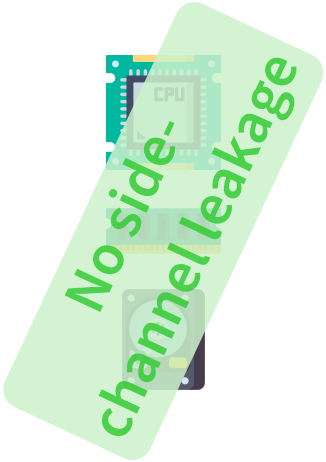
Operating System

Application



Motivation

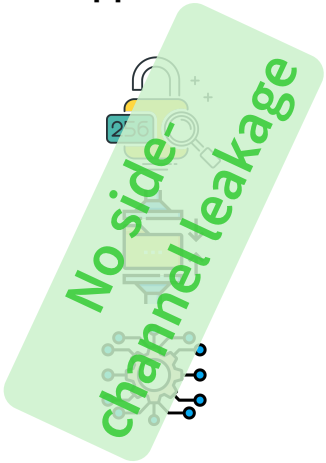
Hardware



Operating System

Research sparse

Application



Linked Lists

Empty list

List with
2 elements

List with
5 elements

Linked Lists

Empty list

List with
2 elements

List with
5 elements

Fast access

Linked Lists

Empty list

List with
2 elements

List with
5 elements

Fast access

Medium fast access

Linked Lists

Empty list

List with
2 elements

List with
5 elements

Fast access

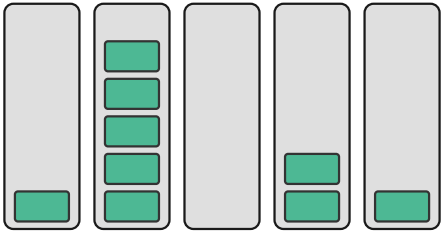
Medium fast access

Slow access

Hash Tables[1]

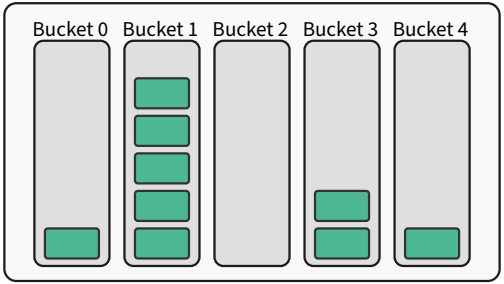
[1] This presentation only covers hash tables. For other data structures, see paper.

Hash Tables[1]



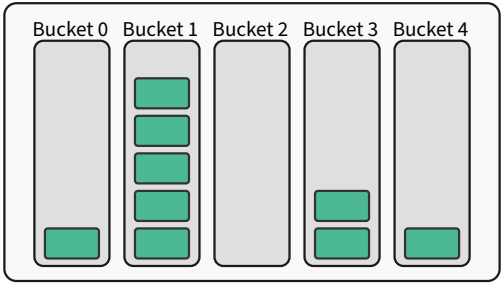
[1] This presentation only covers hash tables. For other data structures, see paper.

Hash Tables[1]



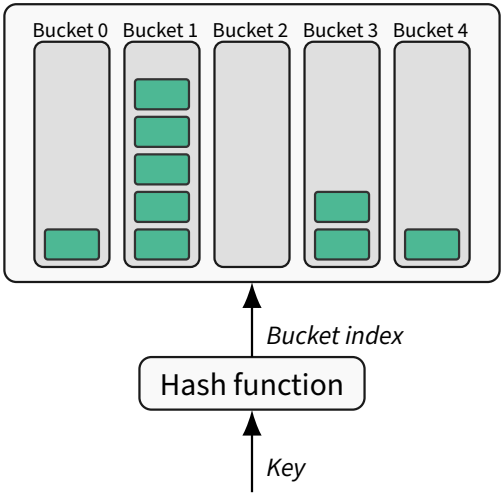
[1] This presentation only covers hash tables. For other data structures, see paper.

Hash Tables[1]



[1] This presentation only covers hash tables. For other data structures, see paper.

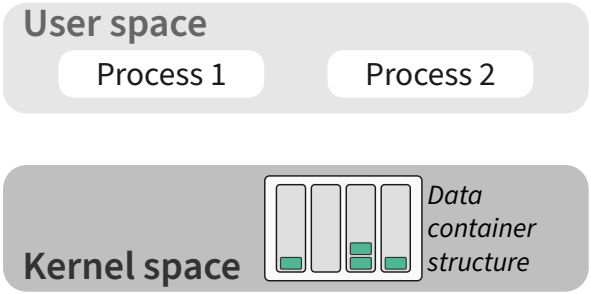
Hash Tables[1]



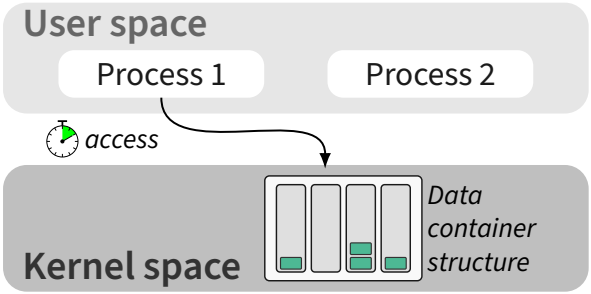
[1] This presentation only covers hash tables. For other data structures, see paper.

KernelSnitch

High-Level Overview

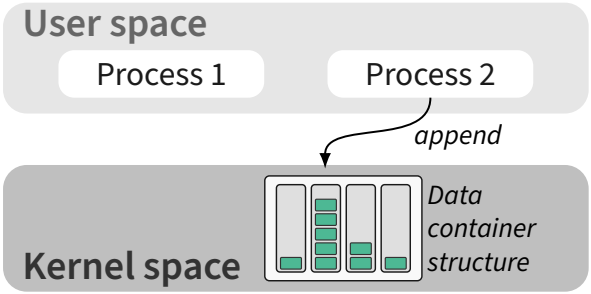


High-Level Overview



Process 1 $\rightarrow$ syscall accesses the data structure

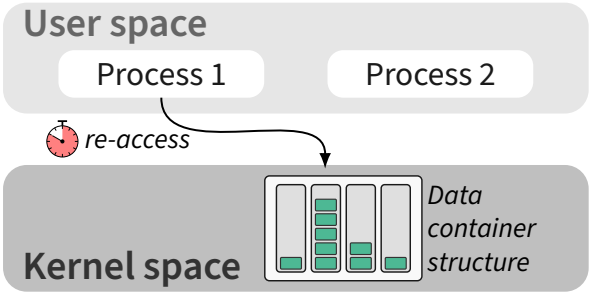
High-Level Overview



Process 1 $\rightarrow$ syscall accesses the data structure

Process 2 $\rightarrow$ syscall appends data

High-Level Overview

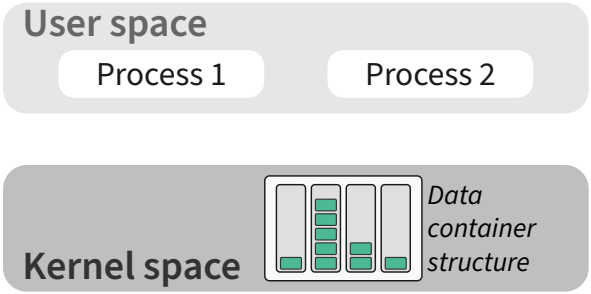


Process 1 $\rightarrow$ syscall accesses the data structure

Process 2 $\rightarrow$ syscall appends data

Process 1 $\rightarrow$ syscall re-accesses the data structure

High-Level Overview



Process 1 $\rightarrow$ syscall accesses the data structure

Process 2 $\rightarrow$ syscall appends data

Process 1 $\rightarrow$ syscall re-accesses the data structure

Deduce security-critical information

Primitives

Primitives

Access primitive

Append/remove primitive

Primitives

Access primitive

- Syscalls that access structures

Append/remove primitive

Primitives

Access primitive

- Syscalls that access structures
- Such as:

Get POSIX time

```
1 sys_clock_gettime(id):
2   sign = current.signal
3   h = timer_hash(id, sign)
4   hbucket =
      posix_timers_htable[h]
5   for tim in hbucket:
6     if tim.sign == sign and
7        tim.id == id:
8       return tim.get_time()
9   return ERROR
```

Append/remove primitive

Primitives

Access primitive

- Syscalls that access structures
- Such as:

Get POSIX time

```
1 sys_clock_gettime(id):
2   sign = current.signal
3   h = timer_hash(id, sign)
4   hbucket =
      posix_timers_htable[h]
5   for tim in hbucket:
6     if tim.sign == sign and
7        tim.id == id:
8       return tim.get_time()
9   return ERROR
```

Append/remove primitive

Hash

Access primitive

- Syscalls that access structures
- Such as:

Get POSIX time

```
1 sys_clock_gettime(id):
2   sign = current.signal
3   h = timer_hash(id, sign)
4   hbucket =
      posix_timers_htable[h]
5   for tim in hbucket:
6     if tim.sign == sign and
7        tim.id == id:
8       return tim.get_time()
9   return ERROR
```

Append/remove primitive

Hash
Bucket

Access primitive

- Syscalls that access structures
- Such as:

Get POSIX time

```
1 sys_clock_gettime(id):
2   sign = current.signal
3   h = timer_hash(id, sign)
4   hbucket =
      posix_timers_htable[h]
5   for tim in hbucket:
6     if tim.sign == sign and
7        tim.id == id:
8       return tim.get_time()
9   return ERROR
```

Hash
Bucket
Timing leak

Append/remove primitive

Access primitive

- Syscalls that access structures
- Such as:

Get POSIX time

```
1 sys_clock_gettime(id):
2   sign = current.signal
3   h = timer_hash(id, sign)
4   hbucket =
      posix_timers_htable[h]
5   for tim in hbucket:
6     if tim.sign == sign and
7        tim.id == id:
8       return tim.get_time()
9   return ERROR
```

Hash
Bucket
Timing leak

Append/remove primitive

- Syscalls that modify structures

Primitives

Access primitive

- Syscalls that access structures
- Such as:

Get POSIX time

```
1 sys_clock_gettime(id):
2   sign = current.signal
3   h = timer_hash(id, sign)
4   hbucket =
      posix_timers_htable[h]
5   for tim in hbucket:
6     if tim.sign == sign and
7        tim.id == id:
8       return tim.get_time()
9   return ERROR
```

Hash
Bucket
Timing leak

Append/remove primitive

- Syscalls that modify structures
- Such as:

Create new POSIX timer

```
1 sys_timer_create():
2   sign = current.signal
3   id = sign.next_id
4   h = timer_hash(id, sign)
5   hbucket =
      posix_timers_htable[h]
6   tim = k_itimer(sign, id)
7   hbucket.append(tim)
8   return id
```

Primitives

Access primitive

- Syscalls that access structures
- Such as:

Get POSIX time

```
1 sys_clock_gettime(id):
2   sign = current.signal
3   h = timer_hash(id, sign)
4   hbucket =
      posix_timers_htable[h]
5   for tim in hbucket:
6     if tim.sign == sign and
7        tim.id == id:
8       return tim.get_time()
9   return ERROR
```

Hash
Bucket
Timing leak

Append/remove primitive

- Syscalls that modify structures
- Such as:

Create new POSIX timer

```
1 sys_timer_create():
2   sign = current.signal
3   id = sign.next_id
4   h = timer_hash(id, sign)
5   hbucket =
      posix_timers_htable[h]
6   tim = k_itimer(sign, id)
7   hbucket.append(tim)
8   return id
```

Primitives

Access primitive

- Syscalls that access structures
- Such as:

Get POSIX time

```
1 sys_clock_gettime(id):
2   sign = current.signal
3   h = timer_hash(id, sign)
4   hbucket =
      posix_timers_htable[h]
5   for tim in hbucket:
6     if tim.sign == sign and
7        tim.id == id:
8       return tim.get_time()
9   return ERROR
```

Hash
Bucket
Timing leak

Append/remove primitive

- Syscalls that modify structures
- Such as:

Create new POSIX timer

```
1 sys_timer_create():
2   sign = current.signal
3   id = sign.next_id
4   h = timer_hash(id, sign)
5   hbucket =
      posix_timers_htable[h]
6   tim = k_itimer(sign, id)
7   hbucket.append(tim)
8   return id
```

Primitives

Access primitive

- Syscalls that access structures
- Such as:

Get POSIX time

```
1 sys_clock_gettime(id):
2   sign = current.signal
3   h = timer_hash(id, sign)
4   hbucket =
      posix_timers_htable[h]
5   for tim in hbucket:
6     if tim.sign == sign and
7        tim.id == id:
8       return tim.get_time()
9   return ERROR
```

Hash
Bucket
Timing leak
Append

Append/remove primitive

- Syscalls that modify structures
- Such as:

Create new POSIX timer

```
1 sys_timer_create():
2   sign = current.signal
3   id = sign.next_id
4   h = timer_hash(id, sign)
5   hbucket =
      posix_timers_htable[h]
6   tim = k_itimer(sign, id)
7   hbucket.append(tim)
8   return id
```

Timing Differences Histogram

Access randomly selected buckets[2]

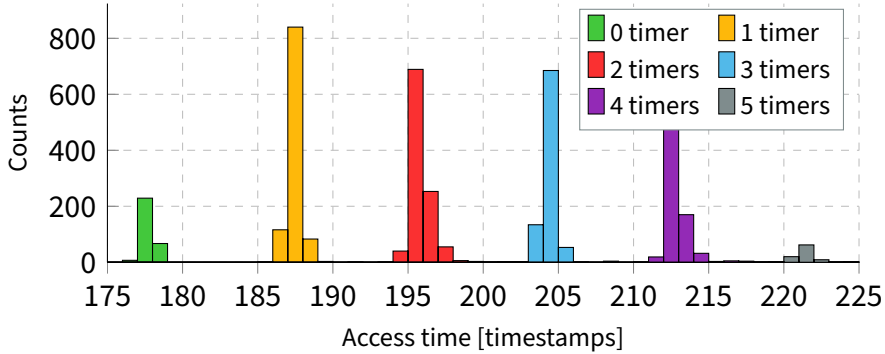
- Repeat 4096 accesses
- Buckets between 0 and 5 timers

[2] Applied information leakage amplification, see paper.

Timing Differences Histogram

Access randomly selected buckets[2]

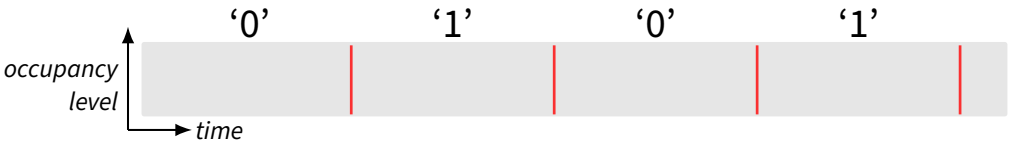
- Repeat 4096 accesses
- Buckets between 0 and 5 timers



[2] Applied information leakage amplification, see paper.

Attacks

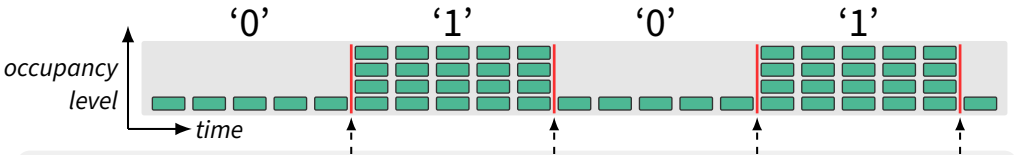
Covert Channel



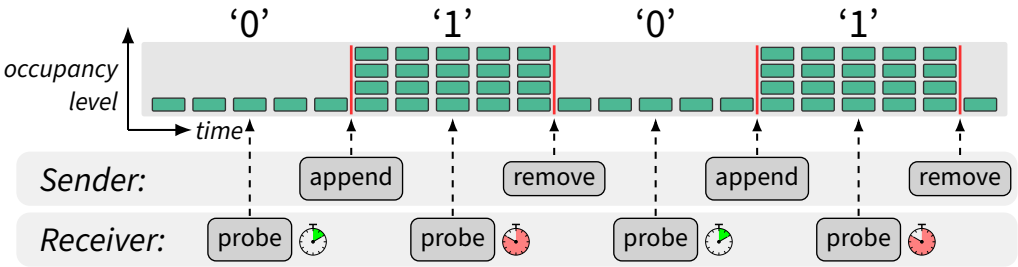
Sender:

Receiver:

Covert Channel



Covert Channel



Performance of Covert Channel

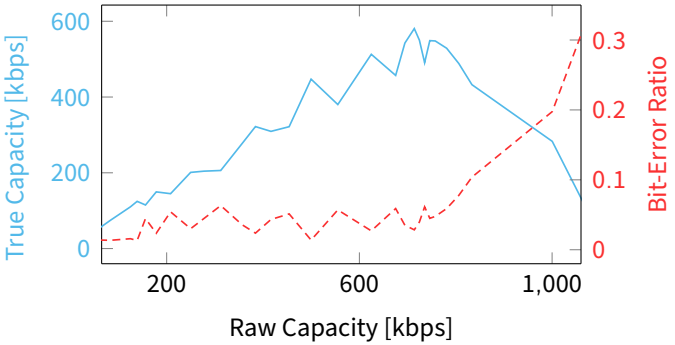
Vary time slice

- Determine raw capacity and BER
- Calculate true capacity

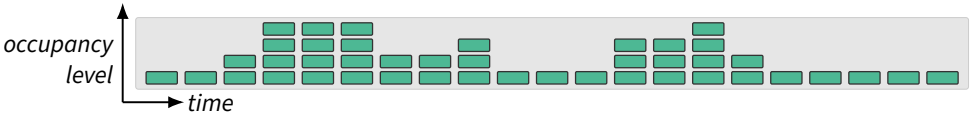
Performance of Covert Channel

Vary time slice

- Determine raw capacity and BER
- Calculate true capacity
- True capacity up to 580 $\mathrm{kbit\,s^{-1}}$



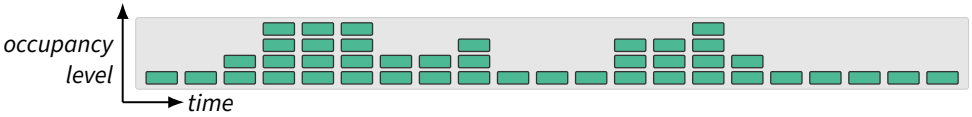
Website Fingerprinting



Browser:

Spy:

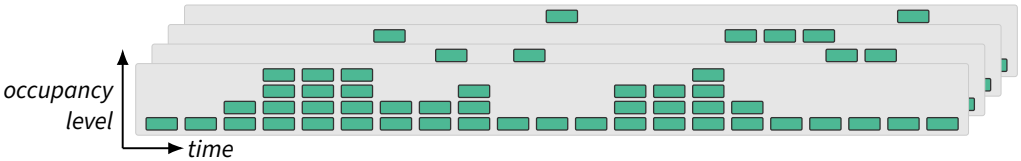
Website Fingerprinting



Browser: Appends/removes when accessing a website ...

Spy:

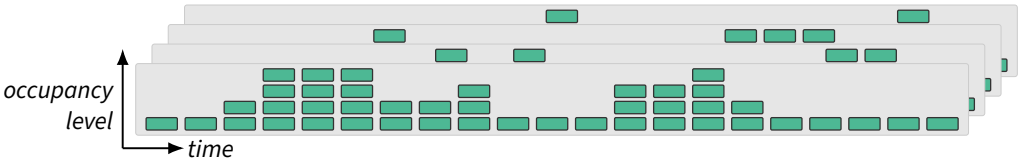
Website Fingerprinting



Browser: Appends/removes when accessing a website for all buckets.

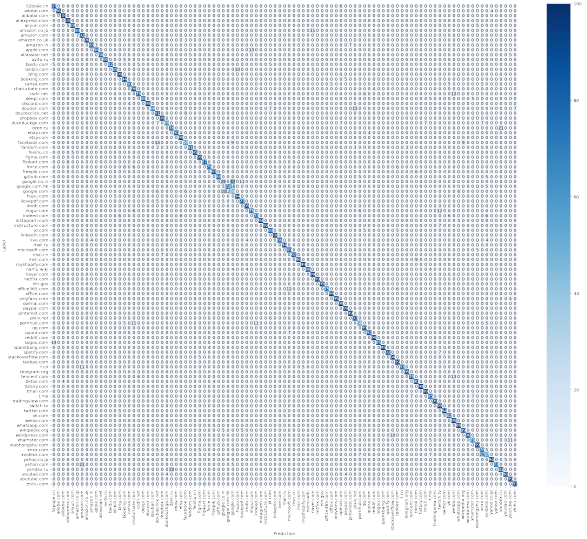
Spy:

Website Fingerprinting



Browser:	Appends/removes when accessing a website for all buckets.
Spy:	Probes all buckets.

Confusion Matrix of Website Fingerprinting



Top 100 websites

- Use Convolutional Neural Network (CNN)
- F_1 score of 89.3 %

Kernel Heap Pointer Leak

Recall:

Access primitive

```
1 index = hash_fn(kaddr, user_id)
2 bucket = hash_table[index]
3 for e in bucket
```

Kernel Heap Pointer Leak

Hash:

- Known: `hash_fn`, `user_id`
- Unkown: `kaddr`

Recall:

Access primitive

```
1 index = hash_fn(kaddr, user_id)
2 bucket = hash_table[index]
3 for e in bucket
```

Kernel Heap Pointer Leak

Recall:

Access primitive

```
1 index = hash_fn(kaddr, user_id)
2 bucket = hash_table[index]
3 for e in bucket
```

Hash:

- Known: `hash_fn`, `user_id`
- Unkown: `kaddr`

Exploit strategy (online phase):

- Same `kaddr`, but different `user_ids`
- Use our side channel to detect hash collisions

Kernel Heap Pointer Leak

Recall:

Access primitive

```
1 index = hash_fn(kaddr, user_id)
2 bucket = hash_table[index]
3 for e in bucket
```

Hash:

- Known: `hash_fn`, `user_id`
- Unkown: `kaddr`

Exploit strategy (online phase):

- Same `kaddr`, but different `user_ids`
- Use our side channel to detect hash collisions

Bruteforce (offline phase):

- Test all possible `kaddrs`
- Match hash collisions

Kernel Heap Pointer Leak of `mm_struct`

Kernel Heap Pointer Leak of `mm_struct`

- Exploit futex hash table
  - `user_id`: user address
  - `kaddr`: `mm_struct`

Kernel Heap Pointer Leak of `mm_struct`

- Exploit futex hash table
  - `user_id`: user address
  - `kaddr`: `mm_struct`
- Consider `mm_struct` alignment
  - $\sim 2^{35.5}$ different kernel addresses

Kernel Heap Pointer Leak of `mm_struct`

- Exploit futex hash table
  - `user_id`: user address
  - `kaddr`: `mm_struct`
- Consider `mm_struct` alignment
  - $\sim 2^{35.5}$ different kernel addresses
- Less than 65 s

Mitigations & Conclusion

Mitigations

Mitigations

- **We** disclosed KernelSnitch.

Mitigations

- **We** disclosed KernelSnitch.
- **Linux Torvalds:** "KASLR is broken for local accesses."
  - No mitigation.

Mitigations

- **We** disclosed KernelSnitch.
- **Linux Torvalds:** "KASLR is broken for local accesses."
  - No mitigation.
- **Kees Cook:** "They leak heap KASLR, not code KASLR. Heap KASLR is hard to expose, even locally."
  - Presented patch.

Mitigations

- **We** disclosed KernelSnitch.
- **Linux Torvalds:** "KASLR is broken for local accesses."
  - No mitigation.
- **Kees Cook:** "They leak heap KASLR, not code KASLR. Heap KASLR is hard to expose, even locally."
  - Presented patch.
- **Linus Torvalds:** "What voodoo programming is this?"
  - Rejected the patch.

Mitigations

- **We** disclosed KernelSnitch.
- **Linux Torvalds:** "KASLR is broken for local accesses."
  - No mitigation.
- **Kees Cook:** "They leak heap KASLR, not code KASLR. Heap KASLR is hard to expose, even locally."
  - Presented patch.
- **Linus Torvalds:** "What voodoo programming is this?"
  - Rejected the patch.
- **Still exploitable to this day!**

Conclusion

Conclusion

- **We presented KernelSnitch**
  - Timing side channel on data structures
  - Software-induced

Conclusion

- **We presented KernelSnitch**
  - Timing side channel on data structures
  - Software-induced
- **We demonstrated leakage amplification**
  - Exploitable from user space
  - From untrusted and isolated users

Conclusion

- **We presented KernelSnitch**
  - Timing side channel on data structures
  - Software-induced
- **We demonstrated leakage amplification**
  - Exploitable from user space
  - From untrusted and isolated users
- **We showed three attacks**
  - Covert channel
  - Website fingerprinting
  - Kernel heap pointer leak

TU Graz
SCIENCE
PASSION
TECHNOLOGY

KernelSnitch

Side-Channel Attacks on Kernel Data Structures

Lukas Maar Jonas Juffinger Thomas Steinbauer Daniel Gruss Stefan Mangard

Graz University of Technology, Austria

NDSS 2025

› isec.tugraz.at