Incentivizing Side-Channel Freedom with Leakage-Aware Computation

Gongqi Huang Leo Chen Amit Levy

Princeton University

Reality 1: Microarchitectural Side-Channels Subvert System Security

Reality 1: Microarchitectural Side-Channels Subvert System Security

CacheBleed (2016)
Spectre (2017)
Meltdown (2017)
Spectre-NG (2018)
Herzbleed (2022)
GoFetch (2024)

Reality 1: Microarchitectural Side-Channels Subvert System Security

CacheBleed (2016)
Spectre (2017)
Meltdown (2017)
Spectre-NG (2018)
Herzbleed (2022)
GoFetch (2024)

...or just `ls /sys/devices/system/cpu/vulnerabilities/`

Reality 2: Infrastructure Providers *Have Little Incentive* to Guarantee Side-Channel Freedom

Reality 2: Infrastructure Providers *Have Little Incentive* to Guarantee Side-Channel Freedom

Requires additional cost.

Reality 2: Infrastructure Providers *Have Little Incentive* to Guarantee Side-Channel Freedom

Requires additional cost.

Performance overhead
Deployment cost
Maintenance cost

Reality 2: Infrastructure Providers *Have Little Incentive* to Guarantee Side-Channel Freedom

When leakage occurs,
Victims **can't** hold providers accountable,

Reality 2: Infrastructure Providers *Have Little Incentive* to Guarantee Side-Channel Freedom

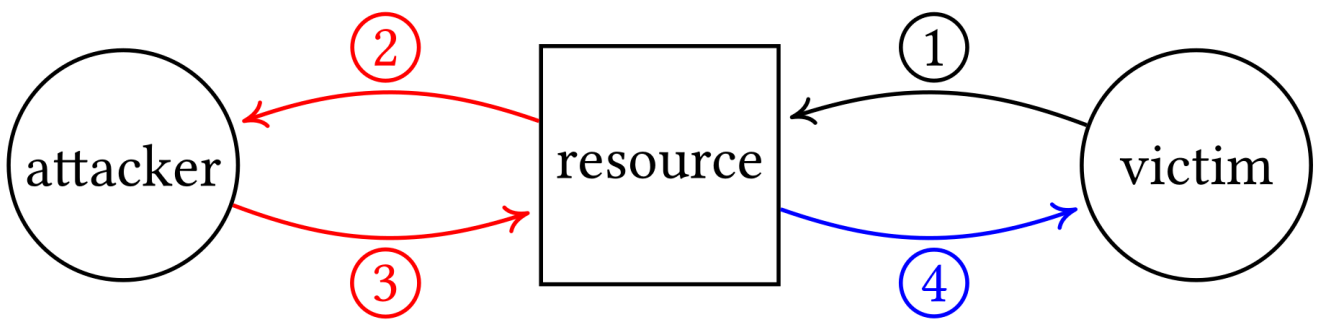
When leakage occurs,
Victims **can't** hold providers accountable,
because µArch leakage is **hard to notice**.

Finding an *ultimate* leak-free system architecture

~~**Finding an *ultimate* leak-free system architecture**~~

Making µArch leakage *tangible* to victims

Symmetry Conjecture of Side Channels



Symmetric Power Between Victim and Attacker

Leakage-Aware Computation

Leakage-Aware Computation

Incentivizes infrastructure providers to close μArch side-channels

Leakage-Aware Computation

Incentivizes infrastructure providers to close μArch side-channels

Enables early-preventive actions for victims (e.g., discard leaked keys)

Leakage-Aware Computation

Incentivizes infrastructure providers to close μArch side-channels

Enables early-preventive actions for victims (e.g., discard leaked keys)

As a starting point, we tackle on *cache* side-channel leakage

Proposal: A program enhancer that converts a program P to a functionally-equivalent *cache*-leakage-aware twin $P^\star$.

Example: P

```
1  if secret {
2    doFoo;
3  } else {
4    doBoo;
5  }
```

Example: $P^{\star}$

```
1 let _  = (*la, *lb, *lc, *ld); // Preload Phase
2 let t0 = now!();               // Execute Phase
3 let t1 = if secret {
4   la: doFoo; lb: now!()
5 } else {
6   lc: doBoo; ld: now!()
7 };
8 if t1 - t0 >= THRES || time!({ let _ = *la; }) >= THRES
9 || ... { panic!("Leaked!"); }  // Check Phase (Omitted
  checks for lb, lc, ld)
```

Example: $P^{\star}$

```
1 let _  = (*la, *lb, *lc, *ld); // Preload Phase
2 let t0 = now!();               // Execute Phase
3 let t1 = if secret {
4   la: doFoo; lb: now!()
5 } else {
6   lc: doBoo; ld: now!()
7 };
8 if t1 - t0 >= THRES || time!({ let _ = *la; }) >= THRES
9 || ... { panic!("Leaked!"); }  // Check Phase (Omitted
  checks for lb, lc, ld)
```

Example: $P^{\star}$

```
1  let _  = (*la, *lb, *lc, *ld); // Preload Phase
2  let t0 = now!();               // Execute Phase
3  let t1 = if secret {
4    la: doFoo; lb: now!()
5  } else {
6    lc: doBoo; ld: now!()
7  };
8  if t1 - t0 >= THRES || time!({ let _ = *la; }) >= THRES
9  || ... { panic!("Leaked!"); }  // Check Phase (Omitted
   checks for lb, lc, ld)
```

Example: $P^{\star}$

```
1 let _  = (*la, *lb, *lc, *ld); // Preload Phase
2 let t0 = now!();               // Execute Phase
3 let t1 = if secret {
4   la: doFoo; lb: now!()
5 } else {
6   lc: doBoo; ld: now!()
7 };
8 if t1 - t0 >= THRES || time!({ let _ = *la; }) >= THRES
9 || ... { panic!("Leaked!"); }  // Check Phase (Omitted
  checks for lb, lc, ld)
```

Example: $P^\star$

```
1  let _  = (*la, *lb, *lc, *ld); // Preload Phase
2  let t0 = now!();               // Execute Phase
3  let t1 = if secret {
4      la: doFoo; lb: now!()
5  } else {
6      lc: doBoo; ld: now!()
7  };
8  if t1 - t0 >= THRES || time!({ let _ = *la; }) >= THRES
9  || ... { panic!("Leaked!"); }  // Check Phase (Omitted
   checks for lb, lc, ld)
```

Questions?