Reproducible research on microarchitectural attacks

Clémentine Maurice, EMSEC

February 12, 2020–Xug meeting, Rennes, France

Reproducible research, it's a good idea, right?

Reproducible research on offensive security?

- Ethical issues if anybody can reproduce attacks without any effort

Reproducible research on offensive security?

- Ethical issues if anybody can reproduce attacks without any effort
- Compromise: some researchers publish the “basic blocks”
  - hope: it’s sort of working, script kiddies can’t use it, experts can modify it for their own research

Reproducible research on offensive security?

- Ethical issues if anybody can reproduce attacks without any effort
- Compromise: some researchers publish the “basic blocks”
  - hope: it’s sort of working, script kiddies can’t use it, experts can modify it for their own research
  - reality: nobody can use it/it depends on how persistent your PhD student is

  $\rightarrow$ some crucial details are left out and there are magic numbers everywhere

Reproducible research on offensive security targeting hardware?

- You can update software, you can't (easily) update hardware

Reproducible research on offensive security targeting hardware?

- You can update software, you can't (easily) update hardware
- Intel knows a thing or two about it

Reproducible research on offensive security targeting hardware?

- You can update software, you can't (easily) update hardware
- Intel knows a thing or two about it
- Our community standards:
  - it's getting better, but so far: if the paper says it runs on two different CPUs that are somewhat recent, we're good!
  - the general sentiment: running code on more than two machines is "just engineering", so we don't care

Reproducible research on offensive security targeting hardware?

- You can update software, you can't (easily) update hardware

Reproducible research on offensive security targeting hardware?

- You can update software, you can't (easily) update hardware
- Intel knows a thing or two about it

Reproducible research on offensive security targeting hardware?

- You can update software, you can't (easily) update hardware
- Intel knows a thing or two about it
- The security community standards:
  - it's getting better, but so far: if the paper says it runs on two different CPUs that are somewhat recent, we're good!
  - the general sentiment: running code on more than two machines is "just engineering", so we don't care

What do I call reproducible

- The same code on the same data works on another machine

What do I call reproducible

- The same code on the same data works on another machine
  - same generation
  - more points to you if different generation/different vendor

What do I call reproducible

- The same code on the same data works on another machine
  - same generation
  - more points to you if different generation/different vendor

Bottom line

1. It's not that easy and it's not "just engineering"
2. I need a pool of heterogeneous machines (mostly different generations)

What do I call reproducible

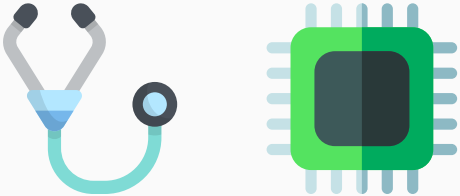
- The same code on the same data works on another machine
  - same generation
  - more points to you if different generation/different vendor

Bottom line

1. It's not that easy and it's not "just engineering"
2. I need a pool of heterogeneous machines (mostly different generations)

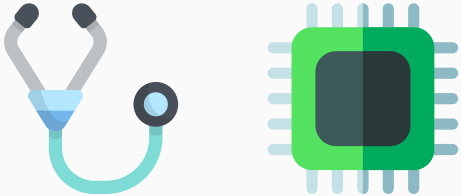
   $\rightarrow$ I hoard laptops forever so I can keep a Sandy Bridge CPU.

What do I do: Side channels on microarchitecture



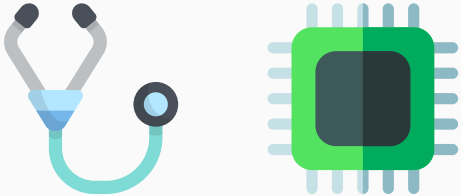
- One axis: finding novel microarchitectural side-channel techniques

What do I do: Side channels on microarchitecture



- One axis: finding novel microarchitectural side-channel techniques
- For example: the cache leaks information, but how you can exploit it depends on some properties of the cache (inclusivity, level of cache targeted…), and of the environment (native code, JS, ARM vs x86…)

What do I do: Side channels on microarchitecture



- One axis: finding novel microarchitectural side-channel techniques
- For example: the cache leaks information, but how you can exploit it depends on some properties of the cache (inclusivity, level of cache targeted…), and of the environment (native code, JS, ARM vs x86…)

→ Basically: find a sequence of instructions that does what I want

Why is it so complicated?

Part I: The Good

a.k.a.
Problems I don’t have

I’m a minimalist

I’m a minimalist

I don’t need:

- fancy clusters
- a lot of memory
- dozens of cores

I’m a minimalist

I don’t need:

- fancy clusters
- a lot of memory
- dozens of cores

→ I usually just use my own laptop to run experiments

People running their experiments on IGRIDA or Grid'5000 be like

Software portability

- The attacks do rely on specific implementations, so if the implementation changes that might be over, but that's fair
- I don't (normally) use fancy features that may change from one OS version to the other, or write code that relies on libraries that will break if the version is not the same

→ Software portability is (mostly) fine

Software portability

- The attacks do rely on specific implementations, so if the implementation changes that might be over, but that's fair
- I don't (normally) use fancy features that may change from one OS version to the other, or write code that relies on libraries that will break if the version is not the same

→ Software portability is (mostly) fine

→ Starts to be an issue when you want to automate things

Part II: The Bad

a.k.a.
Problems I have I can live with

My constraints: sharing is not caring

- No VM $\rightarrow$ messes with timing
- No sharing the hardware $\rightarrow$ would pollute the cache/other microarchitectural component

$\rightarrow$ That's the real reason I typically don't use fancy clusters

My constraints: sharing is not caring

- No VM $\rightarrow$ messes with timing
- No sharing the hardware $\rightarrow$ would pollute the cache/other microarchitectural component

$\rightarrow$ That's the real reason I typically don't use fancy clusters

- Side note: I used Grid'5000 for "Reverse Engineering Intel Last-Level Cache Complex Addressing Using Performance Counters" (RAID 2015)

Part III: The Ugly

a.k.a.
The things that have kept me up many a night

The hardware stack

The nightmare of reproducibility

- Any change in the microarchitecture

The nightmare of reproducibility

- Any change in the microarchitecture
- If it is the same generation, there might be changes in the number of cores, in the size of the caches, associativity…
  - not the end of the world, but requires to have generic code

  → truly engineering: usually okay for your own code, less so if you have code from somebody else with magic values…

The nightmare of reproducibility

- Any change in the microarchitecture
- If it is the same generation, there might be changes in the number of cores, in the size of the caches, associativity…
  - not the end of the world, but requires to have generic code
  - → truly engineering: usually okay for your own code, less so if you have code from somebody else with magic values…
- Roughly one new generation per year, and changes can be quite big
  - that part is the biggest issue

Example #1: Last-level cache complex addressing

Example #2: Cache replacement policy

Reproducing results on another machine might be a scientific contribution (and a top tier paper)

How to run prototypes on several microarchitectures? A girl can dream

- Having a physical platform where one could switch the CPU easily?
  - $\rightarrow$ I frankly have no idea on how to do that

How to run prototypes on several microarchitectures? A girl can dream

- Having a physical platform where one could switch the CPU easily?
  - → I frankly have no idea on how to do that
- What about simulators?

How to run prototypes on several microarchitectures? A girl can dream

- Having a physical platform where one could switch the CPU easily?
  - → I frankly have no idea on how to do that
- What about simulators?
  - simulators like gem5 lack realistic models

How to run prototypes on several microarchitectures? A girl can dream

- Having a physical platform where one could switch the CPU easily?
  - → I frankly have no idea on how to do that
- What about simulators?
  - simulators like gem5 lack realistic models
  - I need realistic models: worthless if the attack only works on the simulator

How to run prototypes on several microarchitectures? A girl can dream

- Having a physical platform where one could switch the CPU easily?
  $\rightarrow$ I frankly have no idea on how to do that

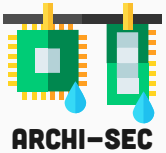
- What about simulators?
  - simulators like gem5 lack realistic models
  - I need realistic models: worthless if the attack only works on the simulator
  - microarchitecture is complex and vastly undocumented $\rightarrow$ reverse-engineering is actually another fun part of my work

How to run prototypes on several microarchitectures? A girl can dream

- Having a physical platform where one could switch the CPU easily?
  → I frankly have no idea on how to do that

- What about simulators?
  - simulators like gem5 lack realistic models
  - I need realistic models: worthless if the attack only works on the simulator
  - microarchitecture is complex and vastly undocumented → reverse-engineering is actually another fun part of my work

  → virtually nobody cares in the community, but I do care (ANR on this topic)



A new hope?

Artifact Evaluation

- Common in some computer science communities, new in security
- ACSAC (since 2017?), USENIX Security (since 2020), WOOT (since 2019)
- Incentive for reproducible research?
- Artifacts are still not part of the evaluation of the paper