Reproducible Research in Micro-architecture Security (and Beyond): from Paper to Artifact Evaluation

Clémentine Maurice, CNRS
@BloodyTangerine

Pass the SALT 2023 keynote - July 4, 2023

Who am I

- ➞ Researcher at CNRS since 2017, currently working at the CRIStAL lab in Lille, France
- ➞ Research in **micro-architectural security**
- ➞ **Member of the “Comité pour la Science Ouverte”** from Ministère de l’Enseignement Supérieur et de la Recherche
- ➞ **Co-chaired multiple Artifact Evaluations**
  - ○ USENIX WOOT’19: first artifact evaluation of the workshop
  - ○ USENIX Security’21 & ‘22: three cycles each
  - ➢ **7 cycles** of artifact evaluation
  - ➢ credit also goes to my co-chairs Alex Gantman, Thorsten Holz, and Cristiano Giuffrida

Outline

1. Reproducible research: **wouldn’t it be great?**
2. **(Personnal) struggles** reproducing micro-architectural security research
3. Artifact Evaluation: **a new hope?**

Reproducible research: wouldn’t it be great?

Imagine...

→ The year is 2023, you want to **compare your method to state of the art**. Authors have open-sourced their code, you compile it, run it, and obtain numbers that you can compare your work with.

Imagine...

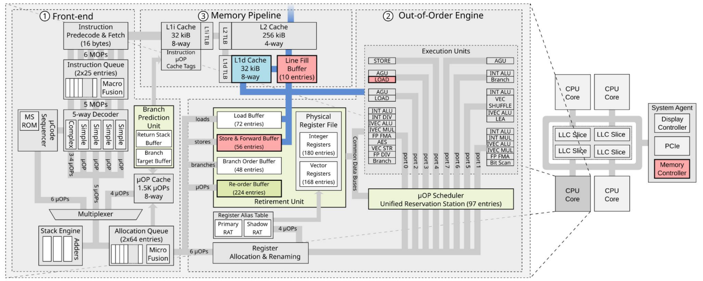
- The year is 2023, you want to **compare your method to state of the art**. Authors have open-sourced their code, you compile it, run it, and obtain numbers that you can compare your work with.
- This is (almost) **science fiction**.

Beyond papers: artifacts

- ➞ A paper is **not just a paper**, it is also a lot of data, code, benchmarks...
- ➞ Problem: it's actually **not trivial to run code in different setups**

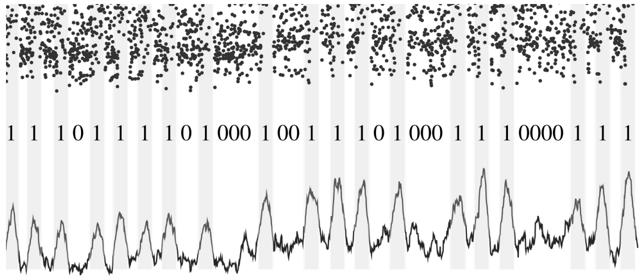
(Personnal) struggles reproducing micro-architectural security research

Micro-architectural security



① Front-end
③ Memory Pipeline
② Out-of-Order Engine
Instruction Predecode & Fetch (16 bytes)
Instruction Queue (2x25 entries)
5-way Decoder
Branch Prediction Unit
µOP Cache 1.5K µOPs 8-way
Multiplexer
Stack Engine
Allocation Queue (2x64 entries)
L1i Cache 32 kiB 8-way
L2 Cache 256 kiB 4-way
L1d Cache 32 kiB 8-way
Line Fill Buffer (10 entries)
Load Buffer (72 entries)
Store & Forward Buffer (56 entries)
Branch Order Buffer (48 entries)
Re-order Buffer (224 entries)
Retirement Unit
Physical Register File
Register Alias Table
Register Allocation & Renaming
Execution Units
µOP Scheduler Unified Reservation Station (97 entries)
CPU Core
LLC Slice
System Agent
Display Controller
PCIe
Memory Controller

→



Hardware usually considered as an abstract layer, but possible attacks:

→ **Fault** attacks: causing **hardware errors** to bypass protections

→ **Side channel** attacks: observing **side effects** of hardware on software execution

Full-software attacks which do not require physical access to hardware

Two sides of the same coin

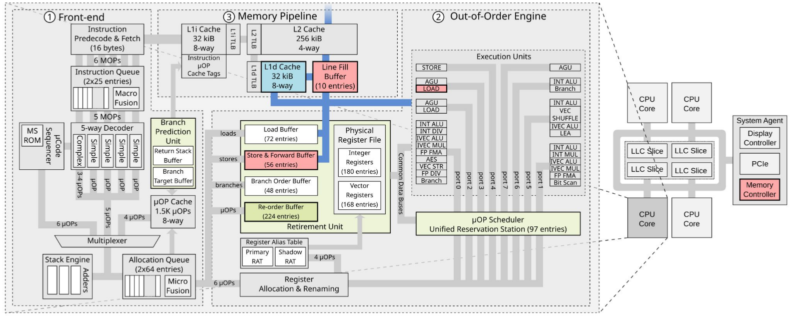
Software implementation

Algorithm 1: Square-and-multiply exponentiation

```
Input: base $b$, exponent $e$, modulus $n$
Output: $b^e \mod n$
$X \leftarrow 1$
for $i \leftarrow bitlen(e)$ downto 0 do
    $X \leftarrow \text{multiply}(X, X)$
    if $e_i = 1$ then
        $X \leftarrow \text{multiply}(X, b)$
    end
end
return $X$
```

&

Hardware



① Front-end
③ Memory Pipeline
② Out-of-Order Engine
Instruction Predecode & Fetch (16 bytes)
L1i Cache 32 kiB 8-way
L2 Cache 256 kiB 4-way
L1d Cache 32 kiB 8-way
Line Fill Buffer (10 entries)
Instruction Queue (2x25 entries)
5-way Decoder
Branch Prediction Unit
µOP Cache 1.5K µOPs 8-way
Load Buffer (72 entries)
Store & Forward Buffer (56 entries)
Branch Order Buffer (48 entries)
Re-order Buffer (224 entries)
Retirement Unit
Physical Register File
Execution Units
µOP Scheduler Unified Reservation Station (97 entries)
Register Alias Table
Register Allocation & Renaming
Allocation Queue (2x64 entries)
Stack Engine
Multiplexer
CPU Core
LLC Slice
System Agent
Display Controller
PCIe
Memory Controller

Research questions

1. Which **software implementations** are vulnerable?
2. Which **hardware components** leak information?

Research questions

1. Which **software implementations** are vulnerable?
2. Which **hardware components** leak information?

Reproducing µ-arch research

- 2015: toward the end of my PhD, I want to reproduce a paper on arXiv on L3 Prime+Probe
- No code but I've been working on cache attacks already and I am confident I can reproduce it
- It **does not work** and **I have no idea why**

Reproducing µ-arch research

- 2015: toward the end of my PhD, I want to reproduce a paper on arXiv on L3 Prime+Probe
- No code but I've been working on cache attacks already and I am confident I can reproduce it
- It **does not work** and **I have no idea why**

Why is it so complicated?

Standards back then

- ➞ If the paper says it runs on two different CPUs that are somewhat recent, we’re good!
- ➞ General sentiment: running code on 2+ machines is “**just engineering**”, so we don’t care
- ➞ Thankfully, **it improved since then**!

Part I: The Good

a.k.a.
Problems I don't have

I am a minimalist

I don’t need:

- fancy clusters
- many cores
- a lot of memory

Most of my experiments can run on my own laptop

Software portability

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 when updated

→ **Software portability is (mostly) fine**

People running their experiments on clusters be like

Part II: The Bad

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

Constraints: sharing is not caring

- No VM → messes with timing
- No sharing the hardware → would pollute the cache/other micro-architectural component
- That’s the real reason I typically don’t use fancy clusters

Part III: The Ugly

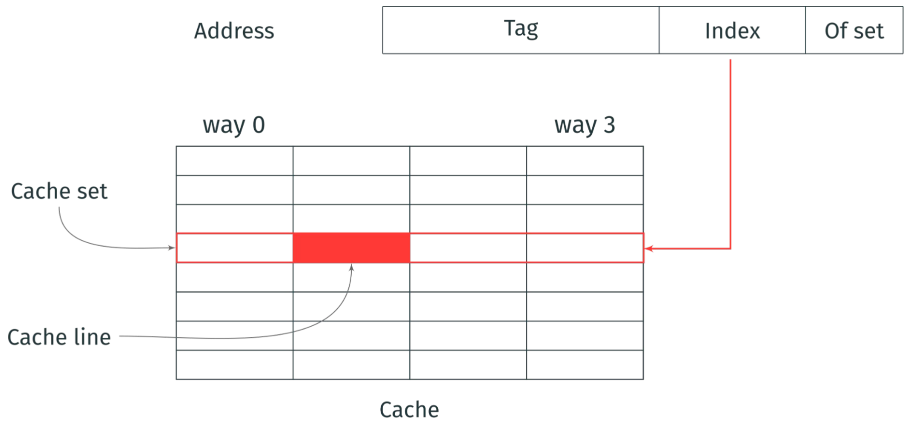
a.k.a. Problems that have kept me up many a night

My actual nightmares

- → **Any change in the micro-architecture**
- → 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**

Let’s get back to Prime+Probe

Set associative caches

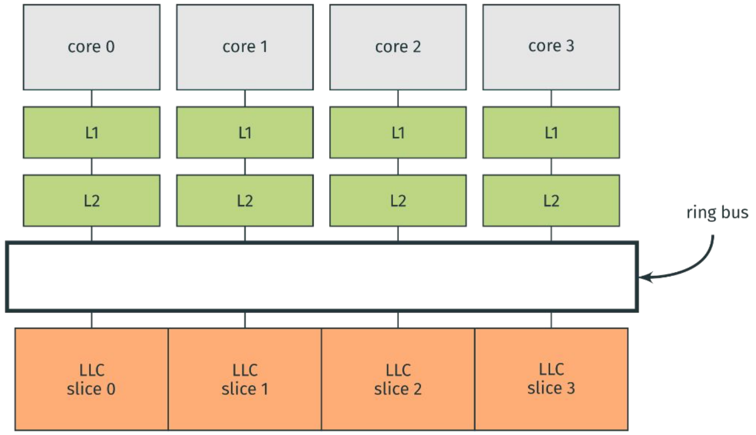


Data loaded in a specific set depending on its address

Several ways per set

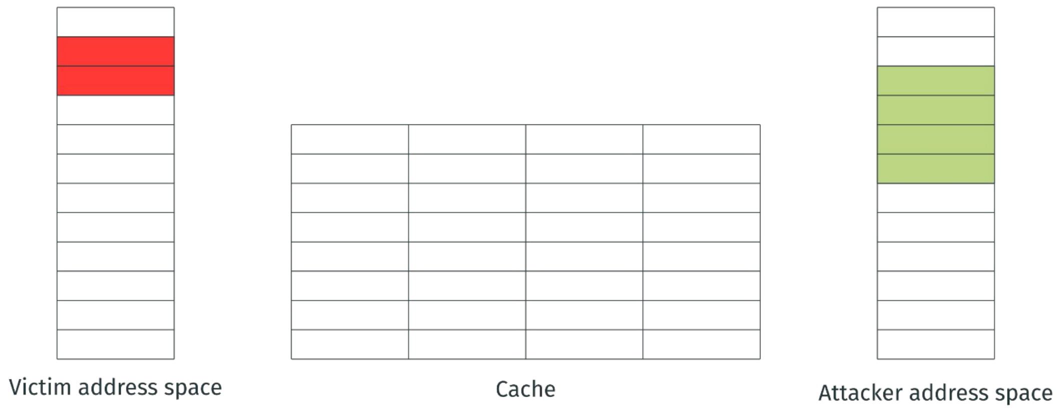
Cache line loaded in a specific way depending on the replacement policy

Caches on Intel CPUs

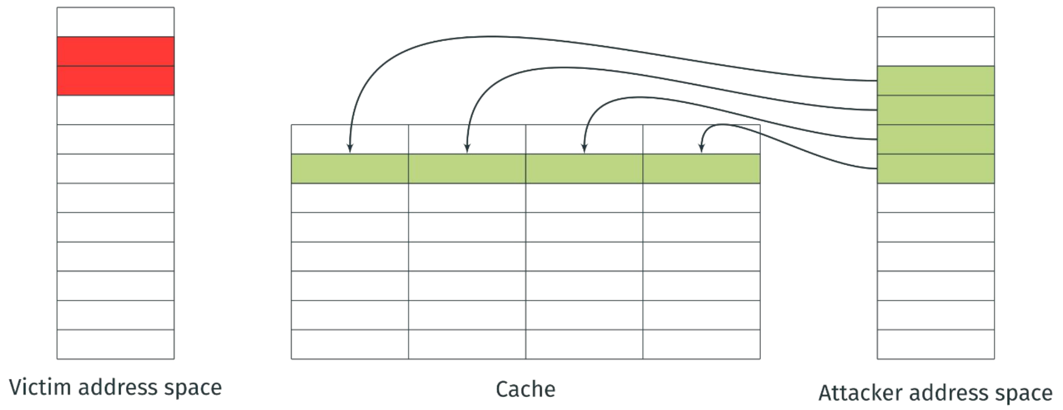


- L1 and L2 are private
- last-level cache
    - divided in slices
    - shared across cores
    - inclusive

Prime+Probe

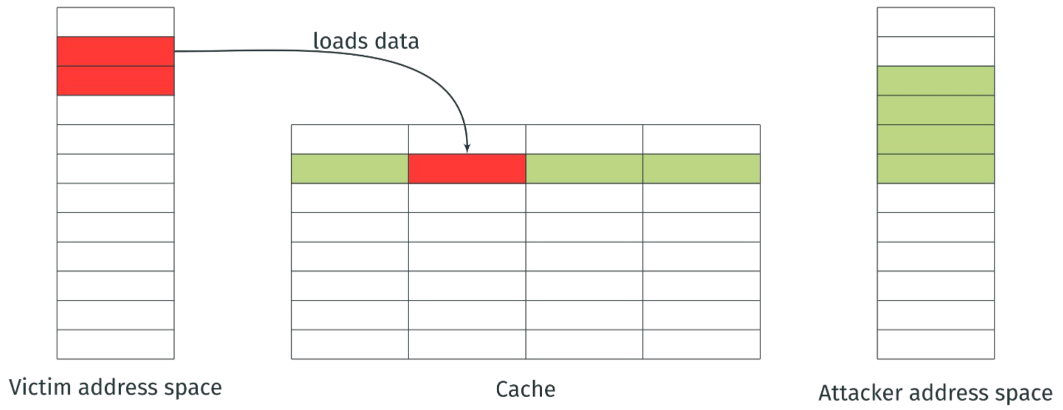


Prime+Probe



Step 1: Attacker primes, *i.e.*, f lls, the cache (no shared memory)

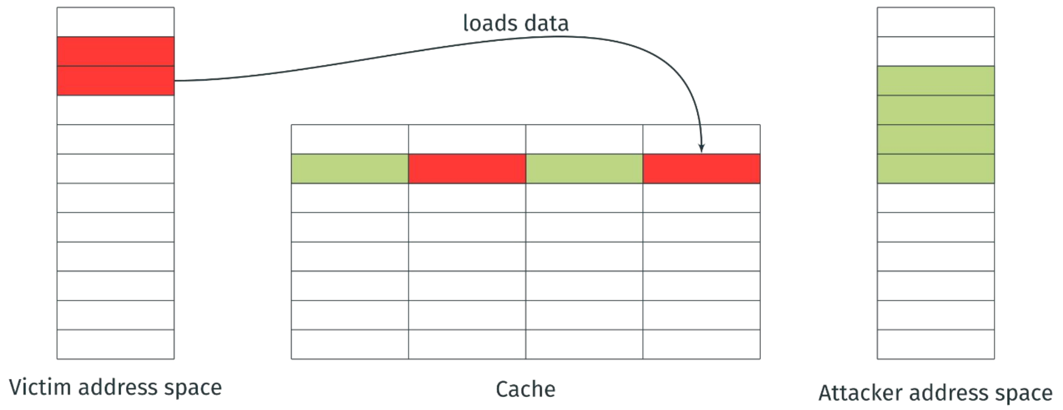
Prime+Probe



Step 1: Attacker primes, *i.e.*, f lls, the cache (no shared memory)

Step 2: Victim evicts cache lines while running

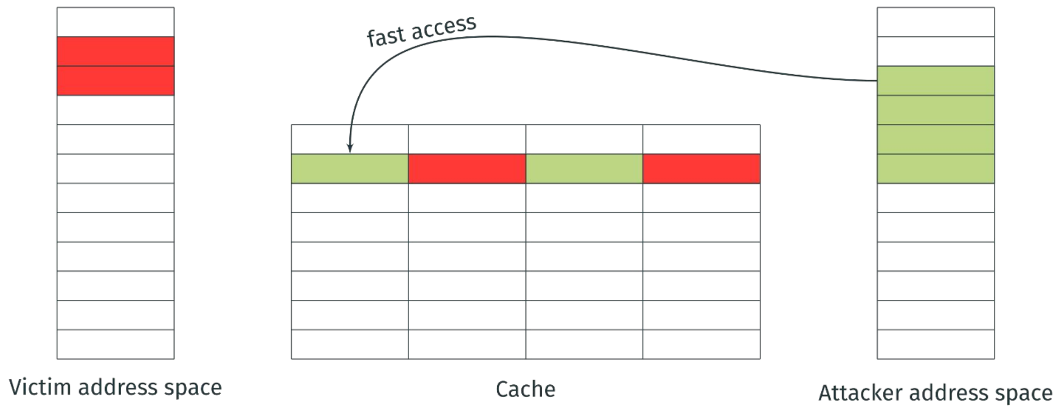
Prime+Probe



Step 1: Attacker primes, *i.e.*, f lls, the cache (no shared memory)

Step 2: Victim evicts cache lines while running

Prime+Probe

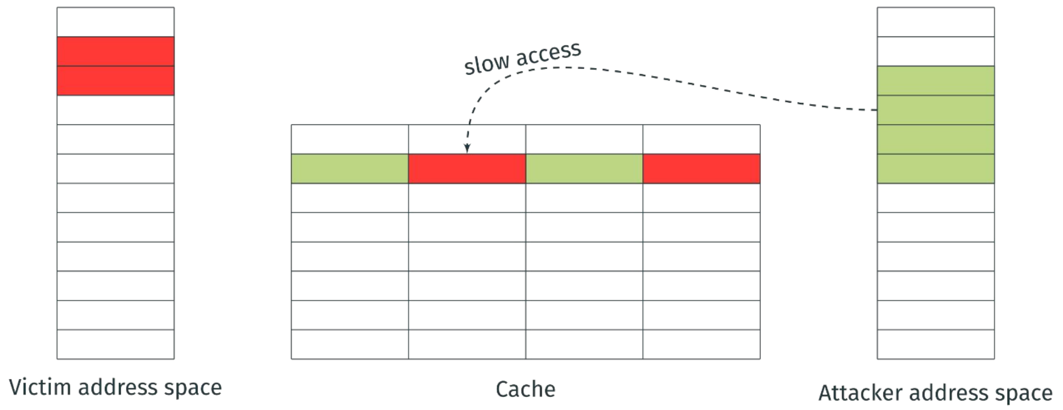


Step 1: Attacker primes, *i.e.*, f lls, the cache (no shared memory)

Step 2: Victim evicts cache lines while running

Step 3: Attacker probes data to determine if set has been accessed

Prime+Probe



Step 1: Attacker primes, *i.e.*, f lls, the cache (no shared memory)

Step 2: Victim evicts cache lines while running

Step 3: Attacker probes data to determine if set has been accessed

Prime+Probe in practice

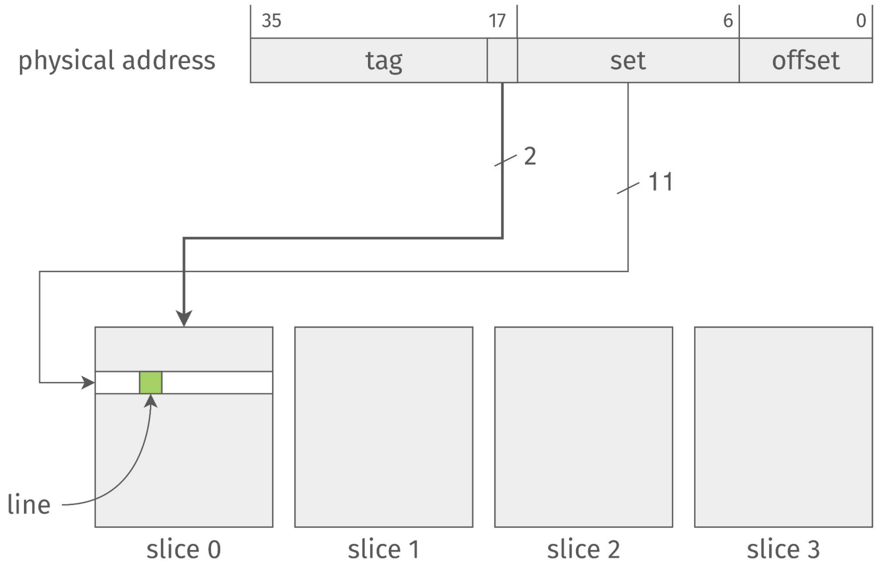
Evicting caches lines without clflush or shared memory:

1. which addresses do we access to have congruent cache lines?
2. without any privilege?
3. and in which order do we access them?

We need:

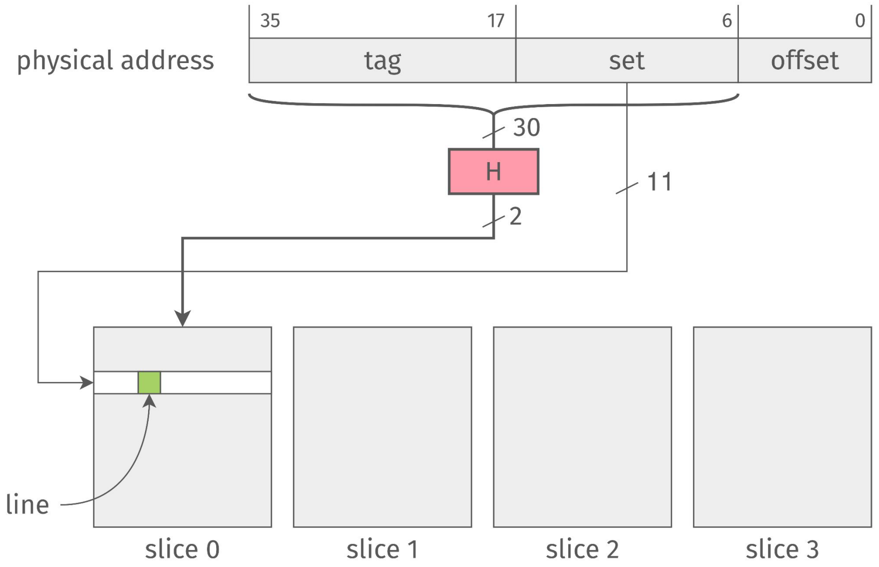
1. an **eviction set**: addresses in the same set, in the same slice (issue #1 and #2)
2. an **eviction strategy** (issue #3)

L3 addressing (before Sandy Bridge)



→ *n* tag bits are used to address the slice

L3 addressing (after Sandy Bridge)

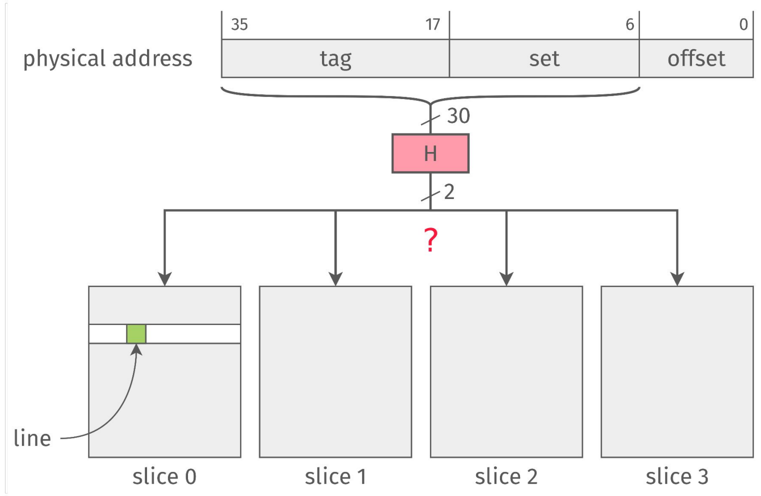


→ complex addressing function is used to address the slice

→ takes as input bits of the set index and tag

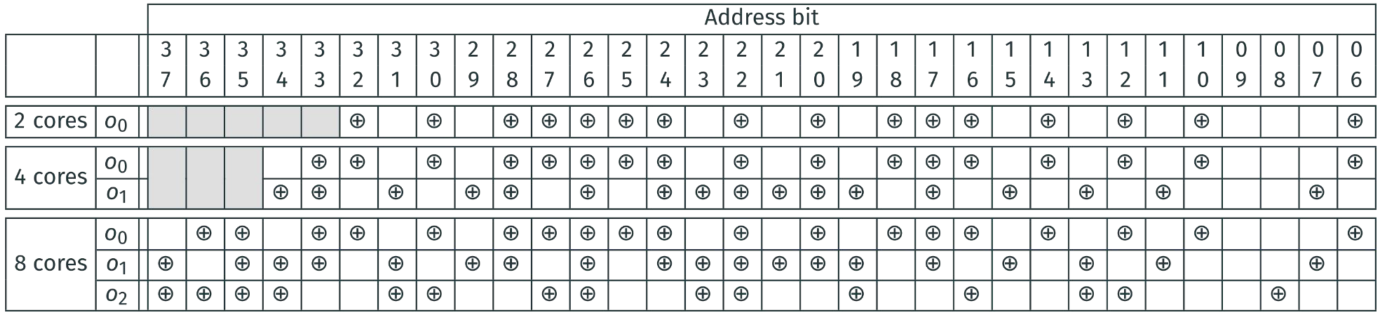
→ **undocumented hash function**

Eviction sets on Sandy Bridge and following



Long story short... here are the functions

3 functions, depending on the number of cores



		Address bit																													
		37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	09	08	07	06
2 cores	o_0	▨	▨	▨	▨	▨	⊕		⊕		⊕	⊕	⊕	⊕	⊕		⊕		⊕		⊕	⊕	⊕		⊕		⊕		⊕				⊕
4 cores	o_0	▨	▨	▨		⊕	⊕		⊕		⊕	⊕	⊕	⊕	⊕		⊕		⊕		⊕	⊕	⊕		⊕		⊕		⊕				⊕
	o_1	▨	▨	▨	⊕	⊕		⊕		⊕	⊕		⊕		⊕	⊕	⊕	⊕	⊕	⊕		⊕		⊕		⊕		⊕				⊕	
8 cores	o_0		⊕	⊕		⊕	⊕		⊕		⊕	⊕	⊕	⊕	⊕		⊕		⊕		⊕	⊕	⊕		⊕		⊕		⊕				⊕
	o_1	⊕		⊕	⊕	⊕		⊕		⊕	⊕		⊕		⊕	⊕	⊕	⊕	⊕	⊕		⊕		⊕		⊕		⊕				⊕	
	o_2	⊕	⊕	⊕	⊕			⊕	⊕			⊕	⊕			⊕	⊕			⊕			⊕			⊕	⊕				⊕		

C. Maurice et al., Reverse Engineering Intel Last-Level Cache Complex Addressing Using Performance Counters. RAID 2015

Reproducing results on another machine might be a scientific contribution

(and a top tier paper)

Artifact Evaluation: a new hope?

Artifact Evaluation

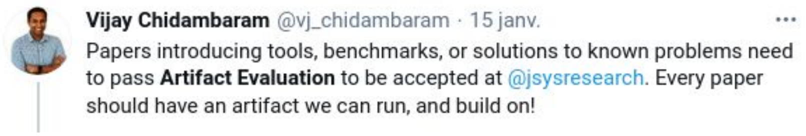
- ➞ Problem: it’s actually **not trivial to run code in different setups**
- ➞ Solution? **Artifact Evaluations**!
  - A group of (**really patient**) people will evaluate the artifact submitted after acceptance of the paper
  - If they can reproduce the results: the paper gets one or several badges

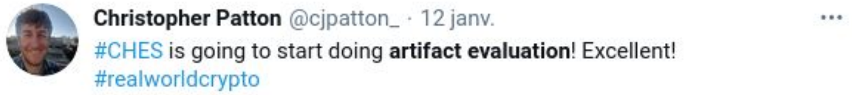
Artifact Evaluation is awesome

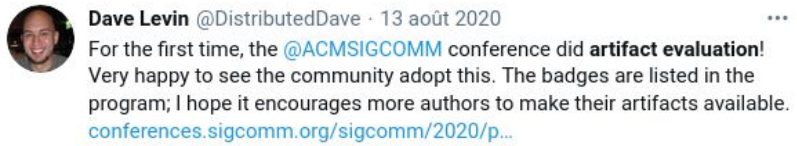
→ **Improving science**: ideally everybody could **replicate** the results to have a higher **confidence** on the paper, **build on it**, and **compare it** with related (passed or future) work

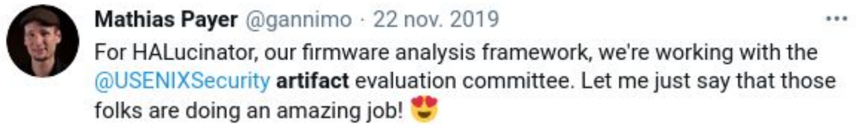
→ Artifact Evaluation is relatively new in security (compared to, e.g., software engineering), but everybody agrees that it is awesome

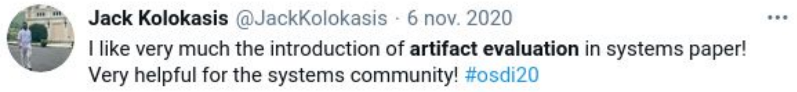
People are very happy about it!


Vijay Chidambaram @vj_chidambaram · 15 janv. ···
Papers introducing tools, benchmarks, or solutions to known problems need to pass **Artifact Evaluation** to be accepted at @jsysresearch. Every paper should have an artifact we can run, and build on!


Christopher Patton @cjpatton_ · 12 janv. ···
#CHES is going to start doing **artifact evaluation**! Excellent! #realworldcrypto


Dave Levin @DistributedDave · 13 août 2020 ···
For the first time, the @ACMSIGCOMM conference did **artifact evaluation**! Very happy to see the community adopt this. The badges are listed in the program; I hope it encourages more authors to make their artifacts available. conferences.sigcomm.org/sigcomm/2020/p...


Mathias Payer @gannimo · 22 nov. 2019 ···
For HALucinator, our firmware analysis framework, we're working with the @USENIXSecurity **artifact** evaluation committee. Let me just say that those folks are doing an amazing job! 😍


Jack Kolokasis @JackKolokasis · 6 nov. 2020 ···
I like very much the introduction of **artifact evaluation** in systems paper! Very helpful for the systems community! #osdi20

Artifact Evaluation process (WOOT & USENIX Security until '22)

"Does the artifact conform to the expectations set by the paper?"

- Authors can submit artifacts **after acceptance** of their paper -- **optional** process
  - They submit: the accepted paper, bidding instructions + sw/hw requirements, and the artifact itself
- AEC members bid on artifacts (so far nobody had more than 1 artifact each session)
- **Discussion phase** between AEC members and authors: ~12 days
  - AEC members are fantastic, this is quite short and makes for an intense phase
- Review phase -- AEC members now have a good idea whether the artifact passed or not: ~ 2 days
- If the paper passed the Artifact Evaluation, the authors add a **badge** before camera ready

Artifact quality

ARTIFACT EVALUATED usenix ASSOCIATION PASSED

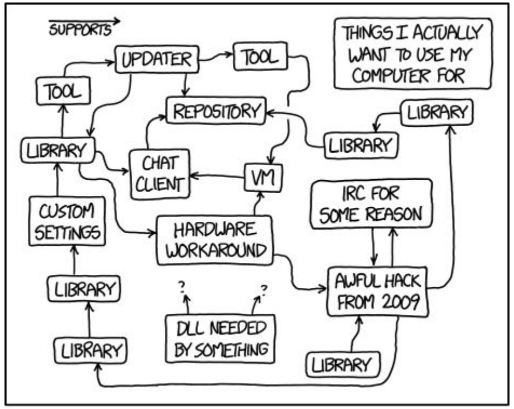
= the artifact conforms to the expectations set by the paper

→ says more about the paper than the artifact, **very variable artifact quality**

Improving artifact quality

Feedback from WOOT '19 AEC members from **what helped or would have helped them**:

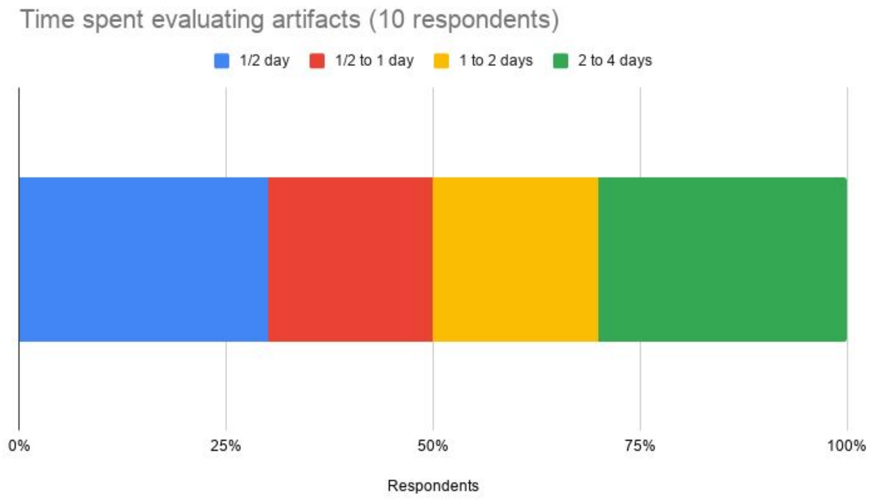
1. Good **documentation**
2. Providing a step-by-step **running example** or automated test cases
3. **Packaging**: VM, docker... anything that avoids Dependency Hell
4. (Providing access to a remote machine)



EVERY NOW AND THEN I REALIZE I'M MAINTAINING A HUGE CHAIN OF TECHNOLOGY SOLELY TO SUPPORT ITSELF.

https://xkcd.com/1579/

Artifact Evaluation is a lot of work

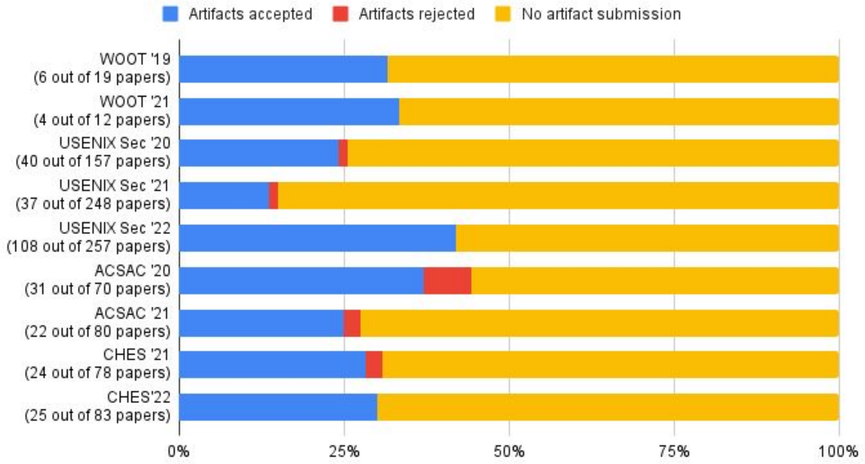


Feedback from WOOT ‘19 AEC

- ➞ Median time: **1 day**, up to 4 days
- ➞ Requires to be **very reactive**
- ➞ Important point: the evaluation is **not adversarial**! AEC members want to make it work!

All the kudos to AEC members!

Artifact Evaluation in the security community



→ **20% to 40%** of accepted papers participated to the Artifact Evaluation

→ That's way less than system conferences! 84% of OSDI '21 accepted papers participated to AE

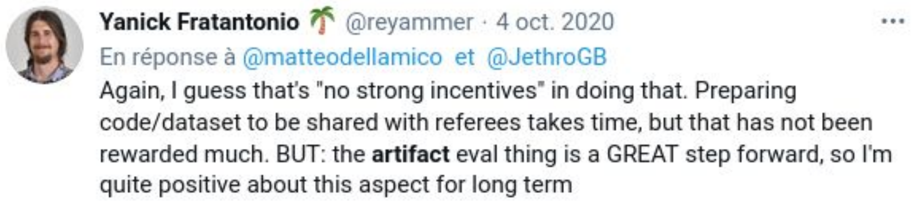
→ No big trend in terms of artifact sharing between workshops and bigger conferences

→ Most submitted artifacts are accepted, most of them are **code**

Caveat of these numbers: only reflect papers gone through the formal evaluation process, not informal sharing

Motivators (1/3)

We collectively agree that Artifact Evaluation Is Awesome, yet less than 30% of papers have an artifact: **what can we do**?


Yanick Fratantonio 🌴 @reyammer · 4 oct. 2020
En réponse à @matteodellamico et @JethroGB
Again, I guess that's "no strong incentives" in doing that. Preparing code/dataset to be shared with referees takes time, but that has not been rewarded much. BUT: the **artifact** eval thing is a GREAT step forward, so I'm quite positive about this aspect for long term

→ We have **limited time** and there are **very little incentives**

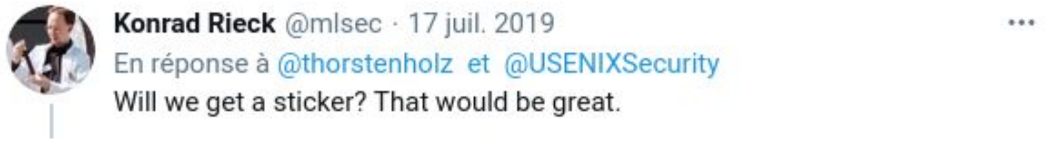
Motivators: short term solutions (2/3)

A very prosaic answer: “**appealing to our inner first graders**”

Motivators: short term solutions (2/3)

A very prosaic answer: "**appealing to our inner first graders**"

STICKERS! Everybody loves stickers!


Konrad Rieck @mlsec · 17 juil. 2019

En réponse à @thorstenholz et @USENIXSecurity

Will we get a sticker? That would be great.

Motivators: long term solutions (3/3)

- ➞ The immense majority of researchers want to do impactful work: **intrinsic motivation**
- ➞ More powerful incentives would not hurt, but **we need to rethink how we evaluate research**
  - ○ Is "number of accepted papers" a good metric? (no, but we already knew that)
  - ○ Can Artifact Evaluations be taken into account in **hiring committees, tenure track committees**?
  - ○ A good start: in our regular evaluations, my employer (CNRS) asks about software production

A few hurdles we experienced

- ➞ **Tight timeline** that has been retrofitted to fit AE, e.g., shepherding and AE at the same time
- ➞ Complicated to **fix hard and fast rules** for all artifacts due to the **diversity**
  - I feel like we run into one or more unexpected questions each AE session
- ➞ Sometimes **only a part** of the paper has a corresponding artifact (for various reasons)
  - Not ideal, but we asked the authors to clarify this in their paper for camera ready

Changes at USENIX Security ‘22

1. More badges!

More complete badges by USENIX (ACM has equivalent badges), already used at OSDI

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE	available for retrieval, permanently and publicly
ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL	documented, completeness, successfully executed
ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED	independently repeatable experiments

Changes at USENIX Security ‘22

2. More time!

→ Past Artifact Evaluations were performed between notification and camera ready

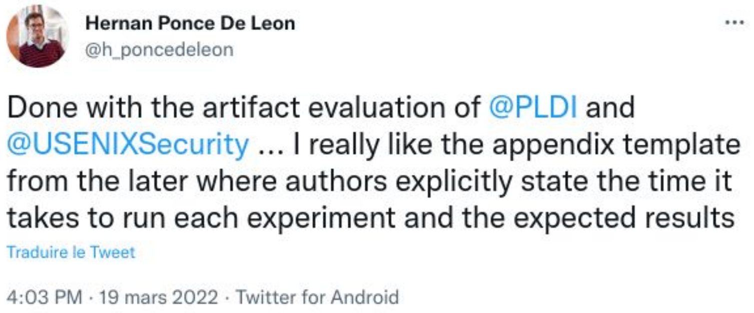
→ Pro: **badges** can be added to the final paper

→ Cons: only leaves around **two weeks of actual evaluation** and very little time for shepherding

→ We are now starting the evaluation **after camera ready**!

Changes at USENIX Security ‘22

3. Unified appendix!



Hernan Ponce De Leon
@h_poncedeleon

Done with the artifact evaluation of @PLDI and @USENIXSecurity … I really like the appendix template from the later where authors explicitly state the time it takes to run each experiment and the expected results

Traduire le Tweet

4:03 PM · 19 mars 2022 · Twitter for Android

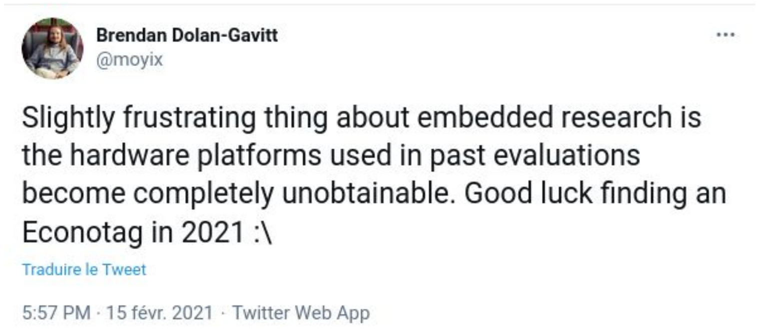
→ Standard Appendix **documenting** the program, dependencies, installation, usage, expected results…

→ Goals: relate **claims** of the paper to the artifact, make it easier to reuse (and to review!)

https://www.usenix.org/conference/usenixsecurity22/artifact-appendix-guidelines

Challenges (1/n)

What about **hardware**?


Brendan Dolan-Gavitt
@moyix

Slightly frustrating thing about embedded research is the hardware platforms used in past evaluations become completely unobtainable. Good luck finding an Econotag in 2021 :\

Traduire le Tweet

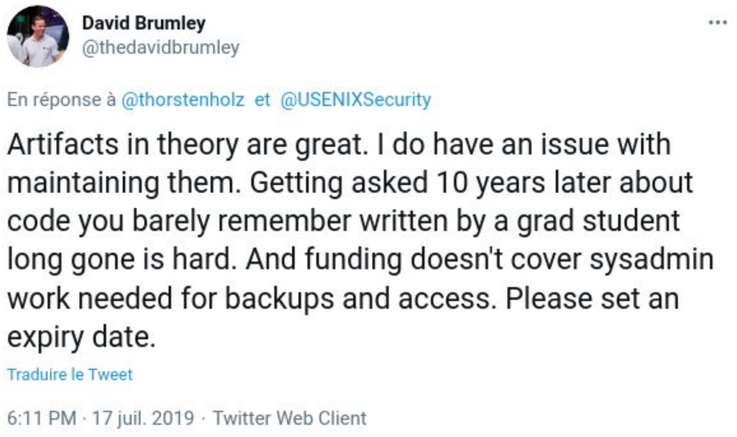
5:57 PM · 15 févr. 2021 · Twitter Web App

→ Hardware **requirements** can be problematic for the evaluation

→ Hardware **availability** will be an issue in a few years

Challenges (2/n)

Actually... what about **software**?



David Brumley
@thedavidbrumley

En réponse à @thorstenholz et @USENIXSecurity

Artifacts in theory are great. I do have an issue with maintaining them. Getting asked 10 years later about code you barely remember written by a grad student long gone is hard. And funding doesn't cover sysadmin work needed for backups and access. Please set an expiry date.

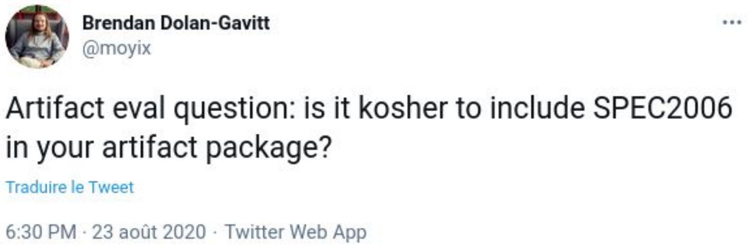
Traduire le Tweet

6:11 PM · 17 juil. 2019 · Twitter Web Client

- → Authors can package beautifully their artifacts to help with software requirements
- → But code probably won't be **maintained** forever
- → Artifact Evaluation probably has a **timestamp**

Challenges (3/n)

Licensing can get in the way of the evaluation


Brendan Dolan-Gavitt
@moyix

Artifact eval question: is it kosher to include SPEC2006 in your artifact package?

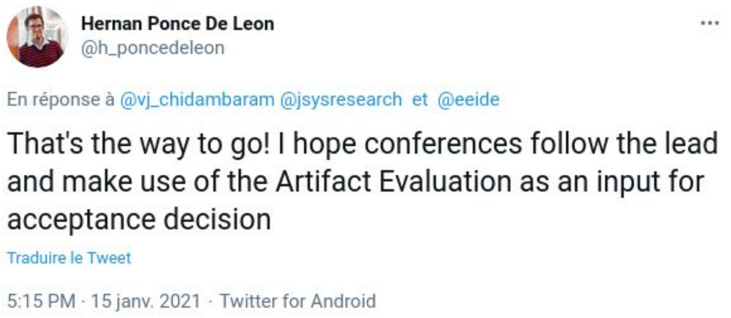
Traduire le Tweet

6:30 PM · 23 août 2020 · Twitter Web App

→ Some artifacts may include **proprietary code**, e.g., SPEC CPU benchmarks are only available for purchase

Challenges (4/n)

It would be great for Artifact Evaluation to **happen during reviews** instead of after acceptance



Hernan Ponce De Leon
@h_poncedeleon

En réponse à @vj_chidambaram @jsysresearch et @eeide

That's the way to go! I hope conferences follow the lead and make use of the Artifact Evaluation as an input for acceptance decision

Traduire le Tweet

5:15 PM · 15 janv. 2021 · Twitter for Android

- → Where to find the **workforce**?
- → ACSAC has opened AE after round 1 of reviews to help decide borderline papers
- → CCS is strongly encouraging authors to provide artifacts at submission time

https://secartifacts.github.io/ is live!

Thanks to Anjo Vahldiek-Oberwagner, Cristiano Giuffrida, Thorsten Holz!

Thank you!