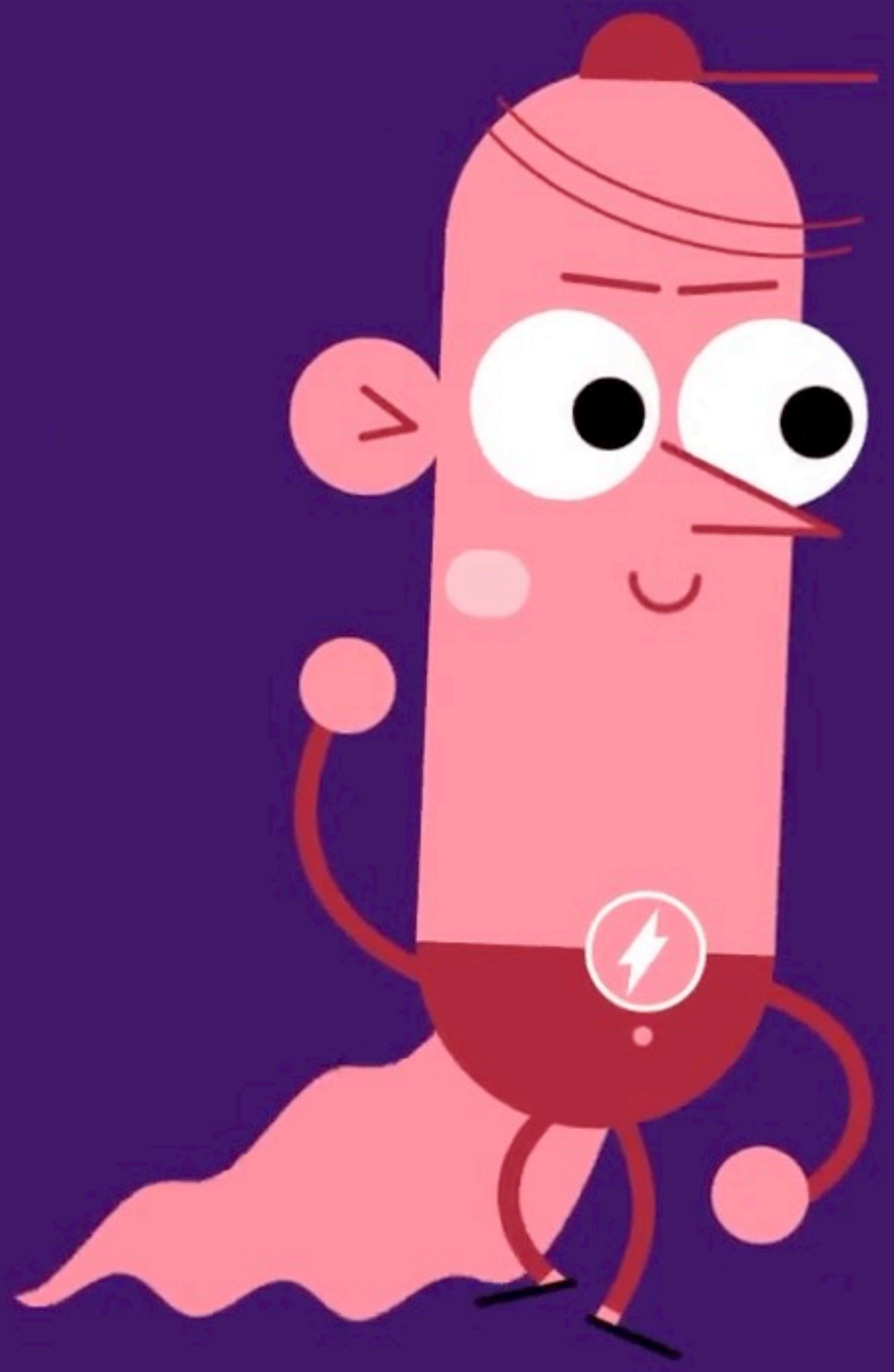
SoK: The Challenges, Pitfalls, and Perils of Using Hardware Performance Counters for Security

Sanjeev Das,

Jan Werner, Manos Antonakakis, Michalis Polychronakis, and Fabian Monrose

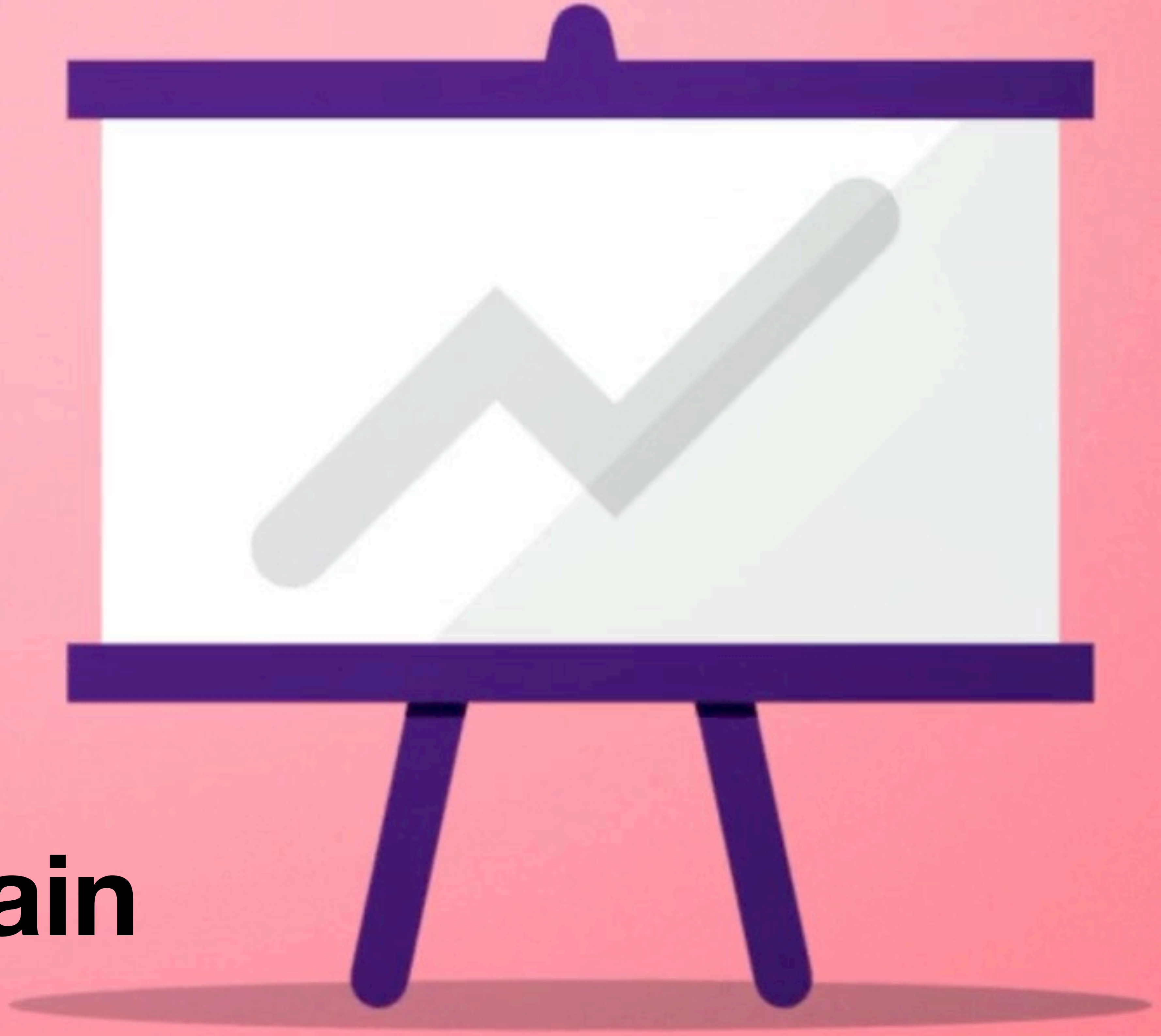
THE UNIVERSITY *of* NORTH CAROLINA *at* CHAPEL HILL

Hardware Performance Counters



- Available in processors for over two decades
- Monitor and measure hardware events, e.g.:
  - Instruction retired, cycles
  - Memory accesses
  - Cache hits/misses
  - Translation look-aside buffer hits/misses

- **Myriad of applications:**
  - **Software Profiling**
  - **Debugging**
  - **High Performance Computing**
  - **Power Analysis**
- **Sharp rise in security domain**



- HPCs provide a good **foundation** for measuring **micro-architectural** information (e.g., branch misses, cache misses)
- Low performance overhead

Recent Security Applications

On the feasibility of **online malware detection** with performance counters. Demme et al., SIGARCH, 2013.

SIGDROP: **Signature-based ROP Detection** using Hardware Performance Counters. Wang et al. [arXiv'16]

Hardware-Assisted Rootkits: Abusing Performance Counters on the ARM and x86 Architectures. Spisak et al. [WOOT'16]

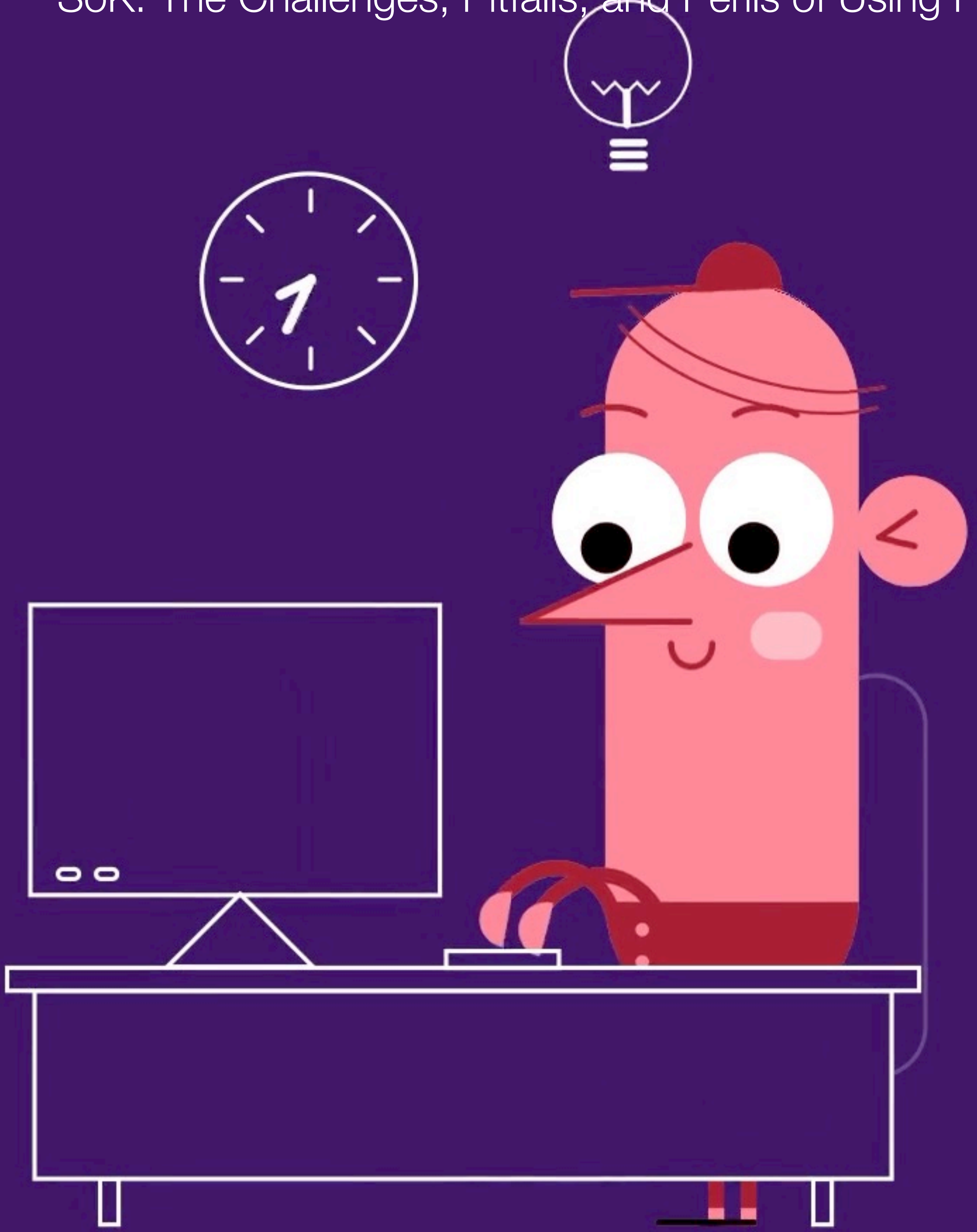
Who Watches the Watchmen?: Utilizing Performance Monitors for **Compromising Keys of RSA on Intel Platforms**, Bhattacharya et al.[CHES'15]

Recent Security Applications

Detecting Spectre And Meltdown Using Hardware Performance Counters. Pierce, Endgame Inc., Jan. 08, **2018**

Detecting Attacks that Exploit Meltdown and Spectre with Performance Counters. Fiser & Gamazo Sanchez, Trend Micro Inc., **2018**

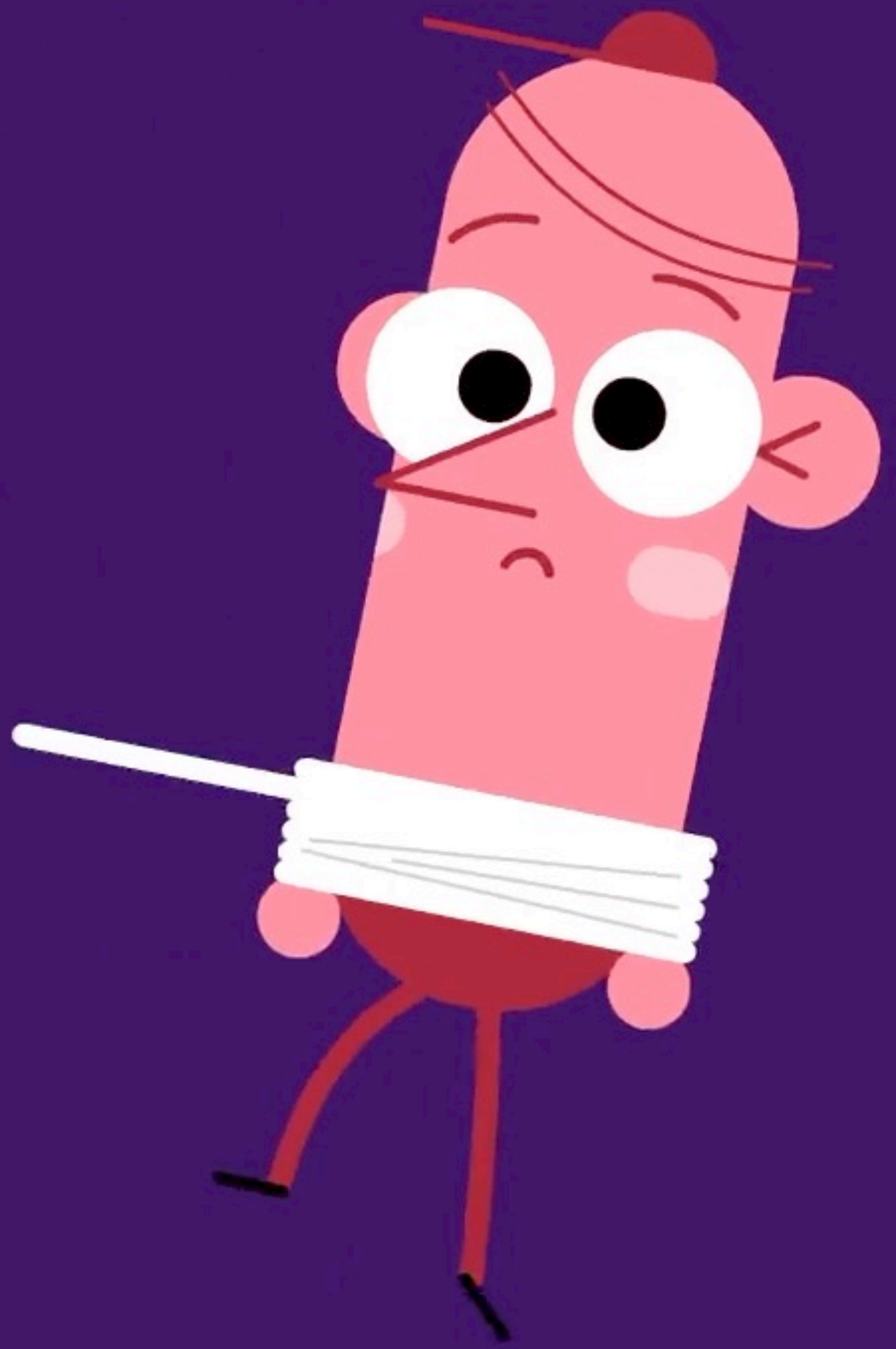
Detecting Spectre Attacks by identifying Cache Side-Channel Attacks using Machine Learning. Depoix et al. [WAMOS, **2018**]



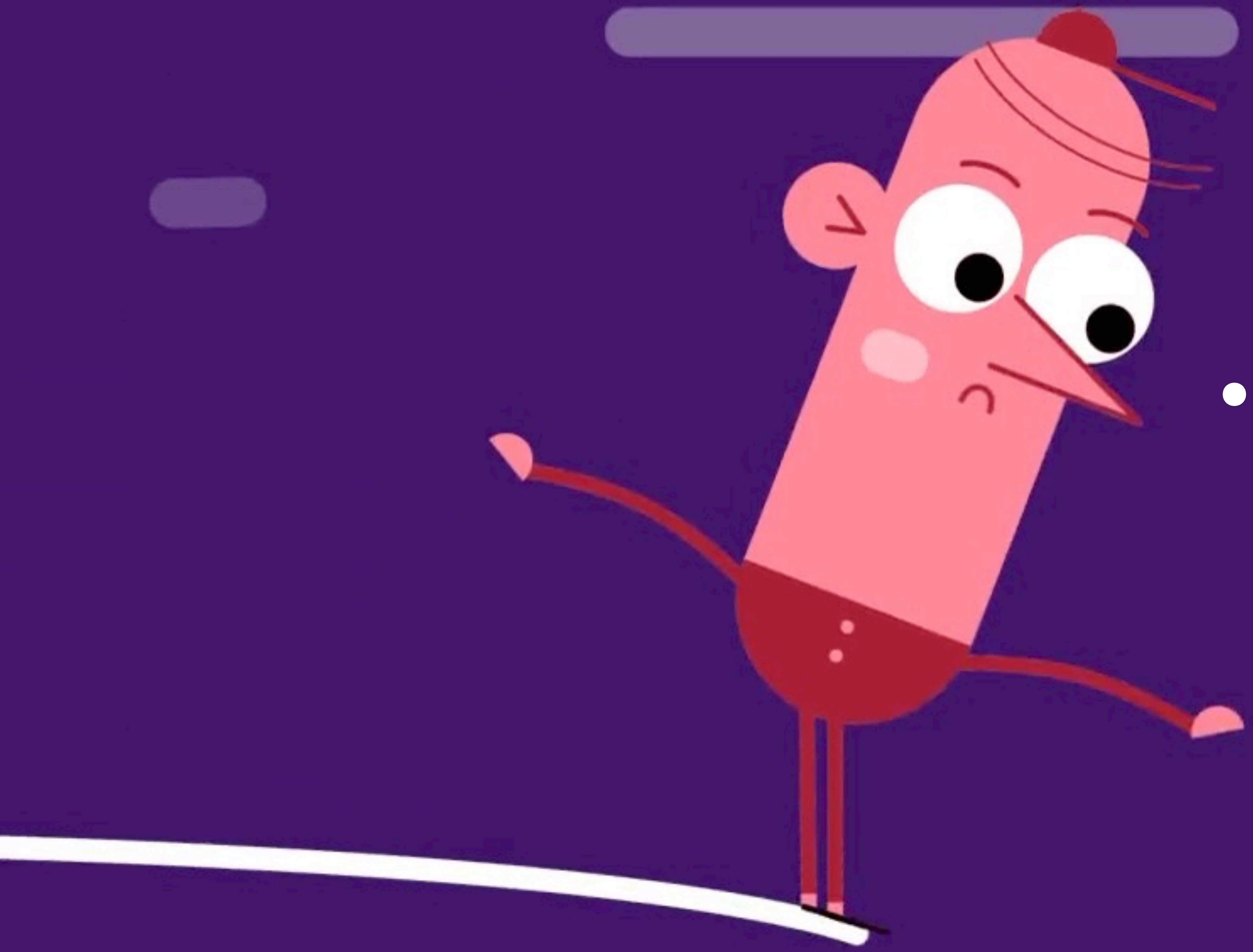
Impetus of this SoK paper:

Can we use HPCs as a foundation for thwarting Data Only Attacks?

Challenges



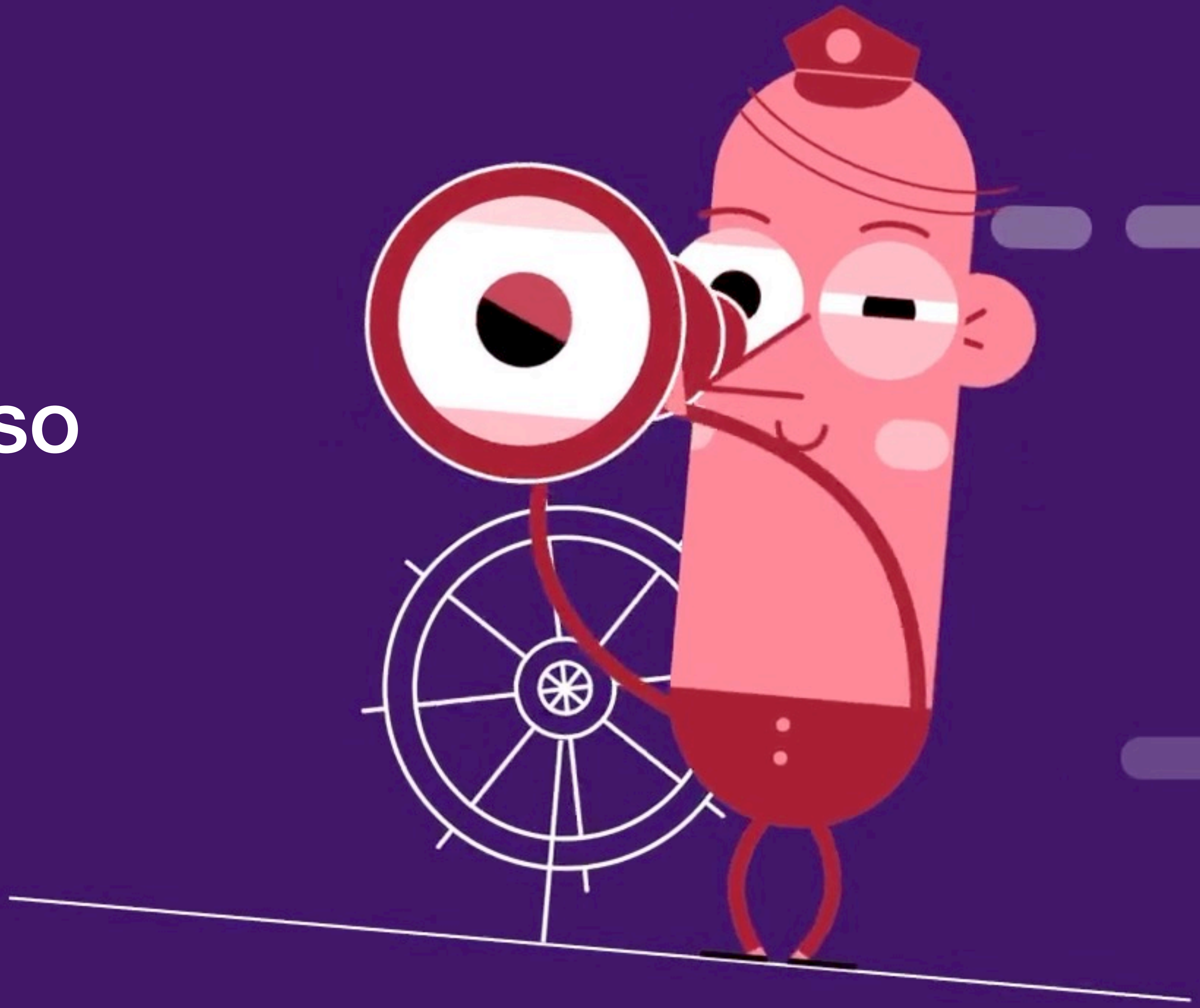
- Which events should we measure?
  - There are **HUNDREDS** of HPC events
  - How are the events related to each other?
- Is there a standard way to collect HPC measurements?
- What framework should we use?
  - Collection techniques vary widely



- **Non-determinism** issue in HPCs
  - ***"Can hardware performance counters be trusted?"*** Weaver & McKee, Workload Characterization, 2008

- Lack of **application-level** profiling
  - No process-level filtering of HPC data at the hardware level

Did other researchers also notice these pitfalls?



- We analyzed nearly **100** papers from different application domains

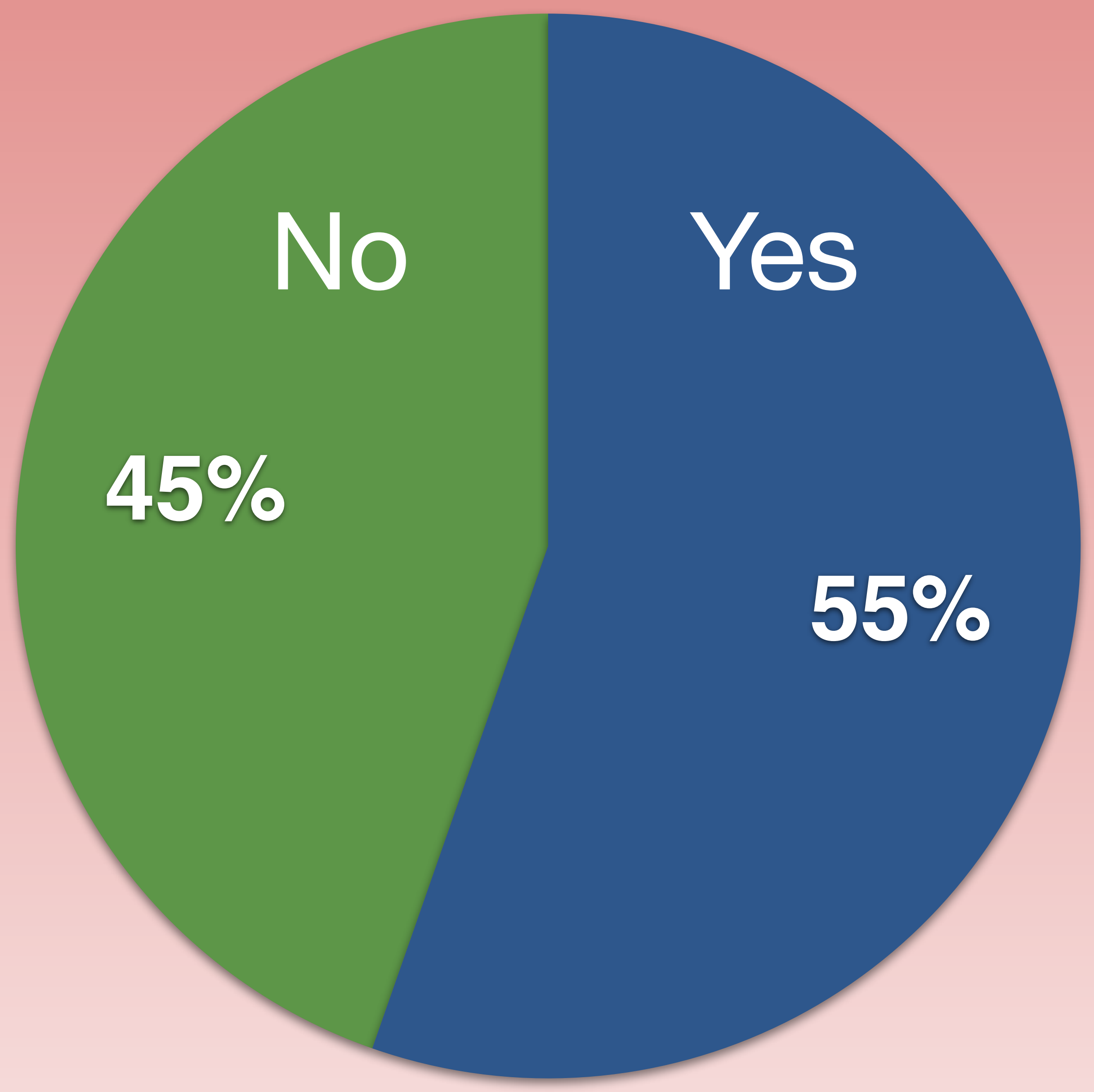
- Debugging
- Power Analysis
- Performance Analysis
- Security

- We also conducted a survey:
  - Sent questionnaire to authors
  - After repeated attempts, response was 28%

Findings

- We examined 56 papers that acknowledged non-determinism issues from non-security application domains
- Painstakingly evaluated if they recommended using HPCs
  - 45% of the papers did not, because of lack of determinism and portability

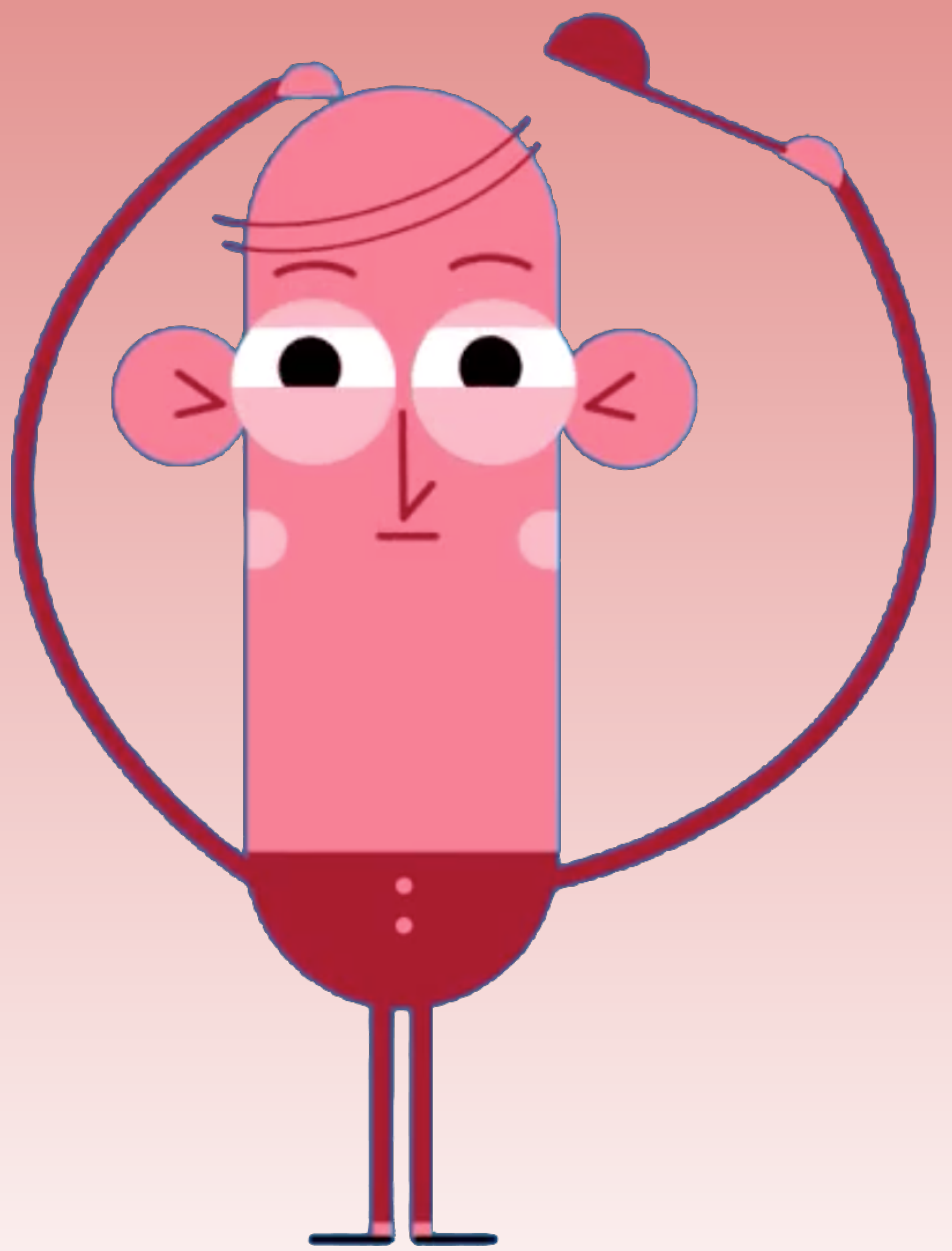
Non-security domains



Findings

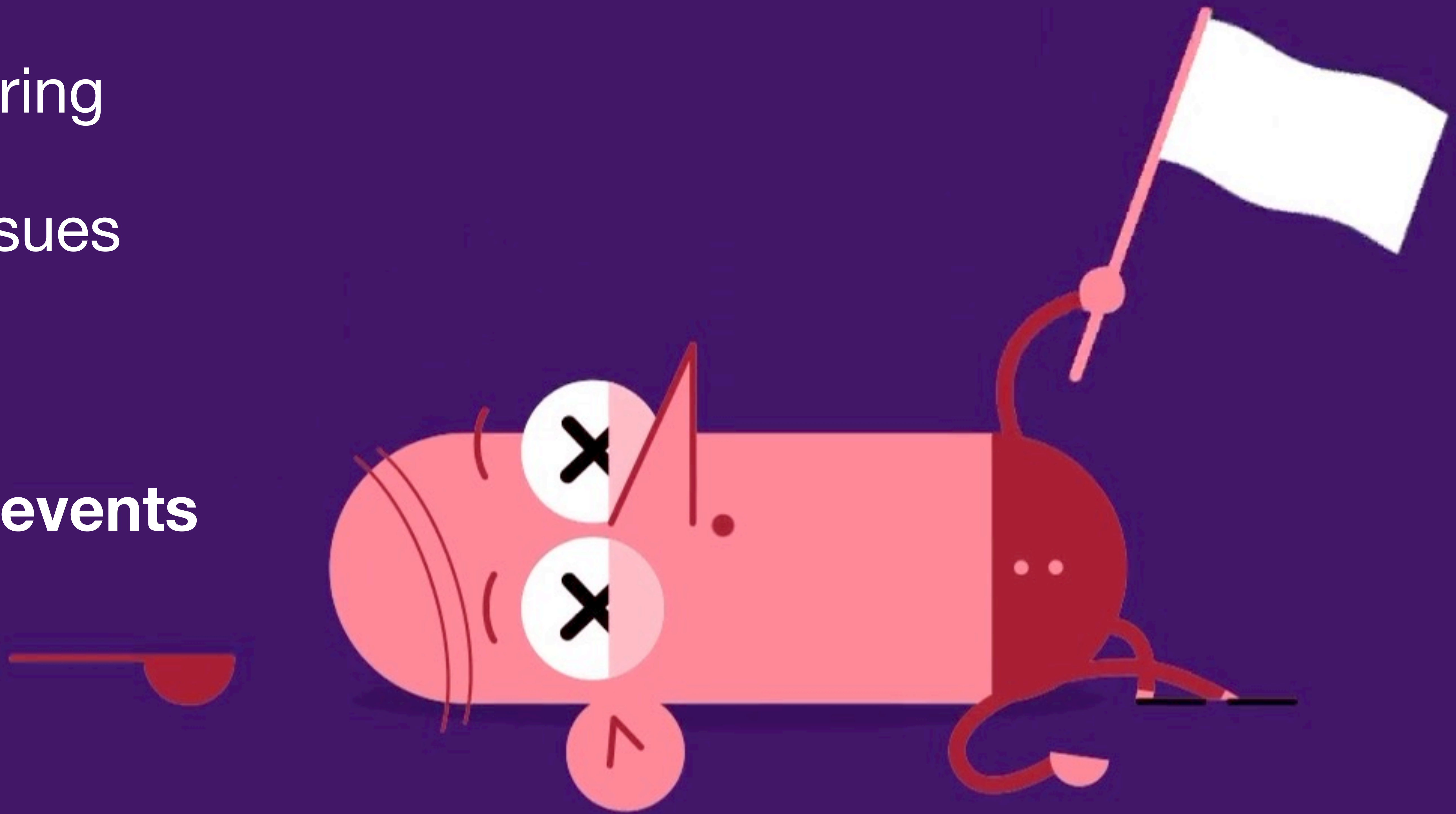
- Of the 40 security papers that used HPCs
  - Only 10% acknowledge non-determinism issues
  - Acceptance of HPCs in security is in stark contrast to other domains

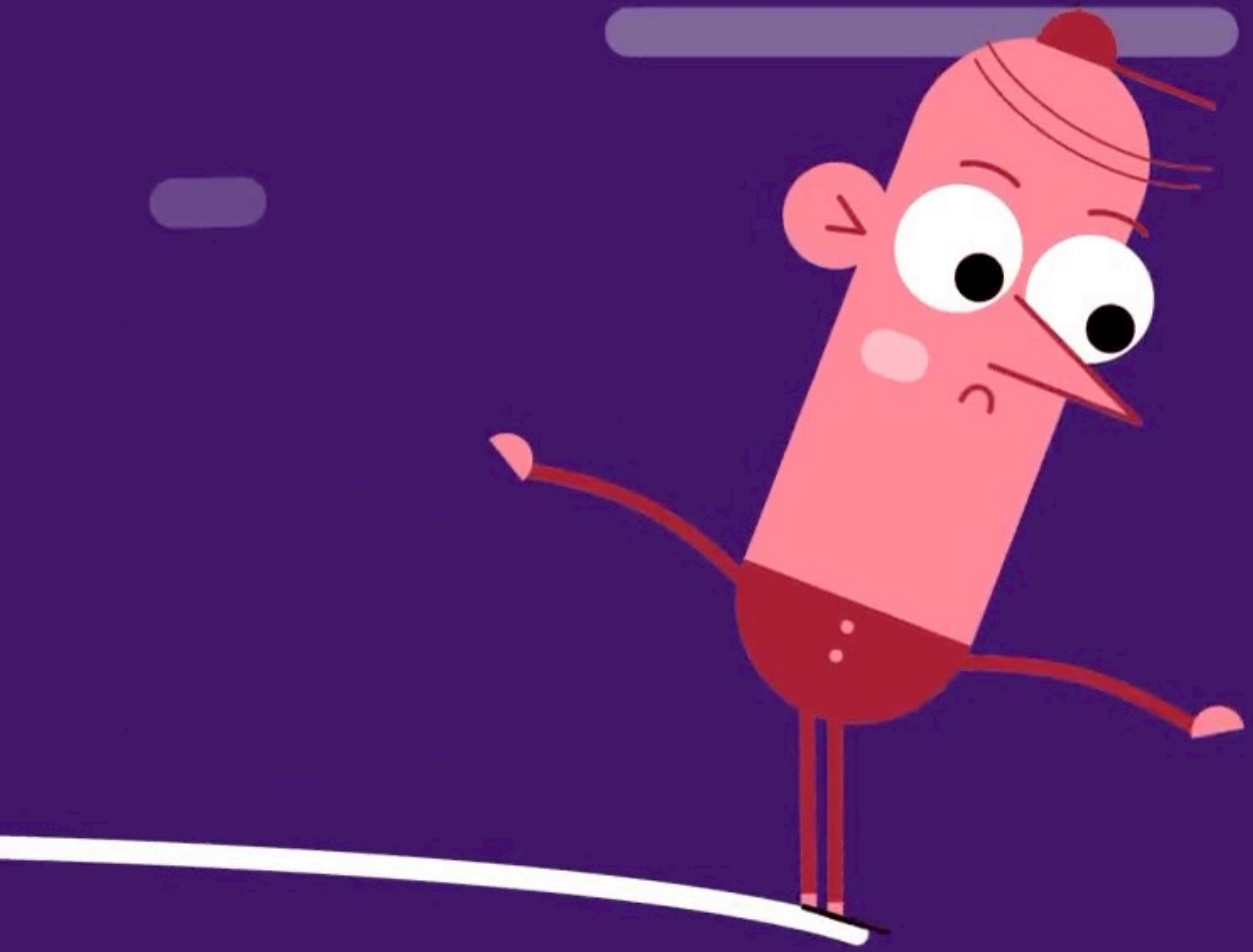
Can hardware performance counters be trusted?
Weaver & McKee, Workload Characterization, 2008



Common Failures

- **Mishandling** of performance counter data
  - Lack of process-level filtering
- Ignoring non-determinism issues
  - **Skid**
  - **Over/under-counting of events**





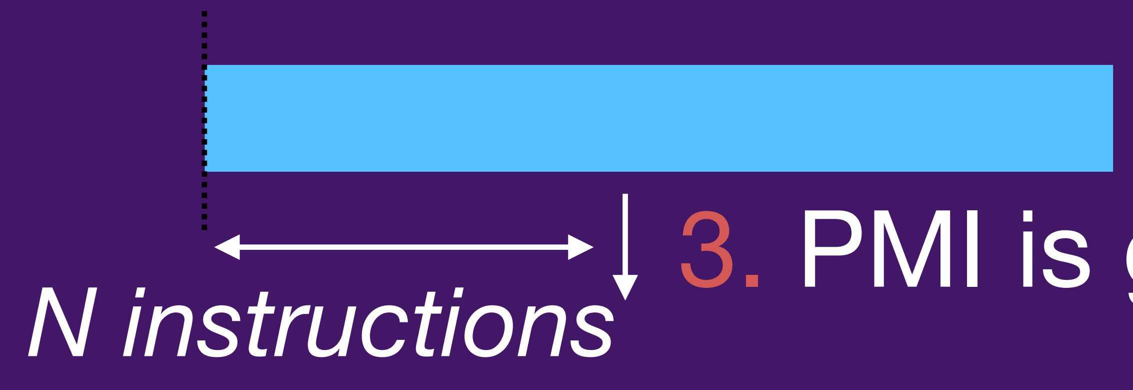
Handling of HPC Data

- Limited number of programmable counters
- Configuration
  - done in kernel mode by reading and writing into model specific registers (MSRs)
  - Two modes : **Polling** vs **Sampling**

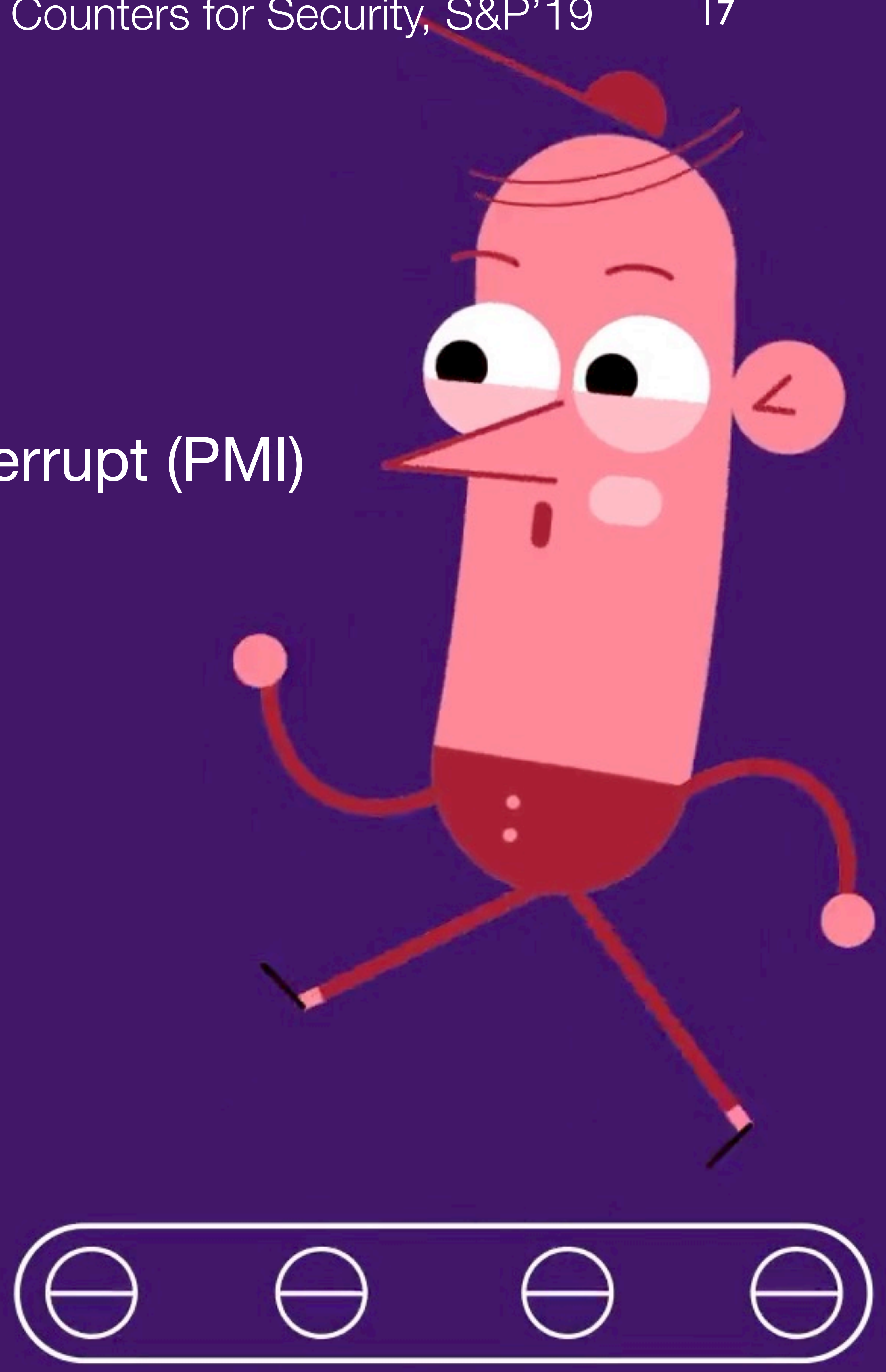
Handling of HPC Data

Event-based sampling using Performance Monitoring Interrupt (PMI)

1. Configure events in sampling mode, e.g., N instructions retired
2. Program begin execution

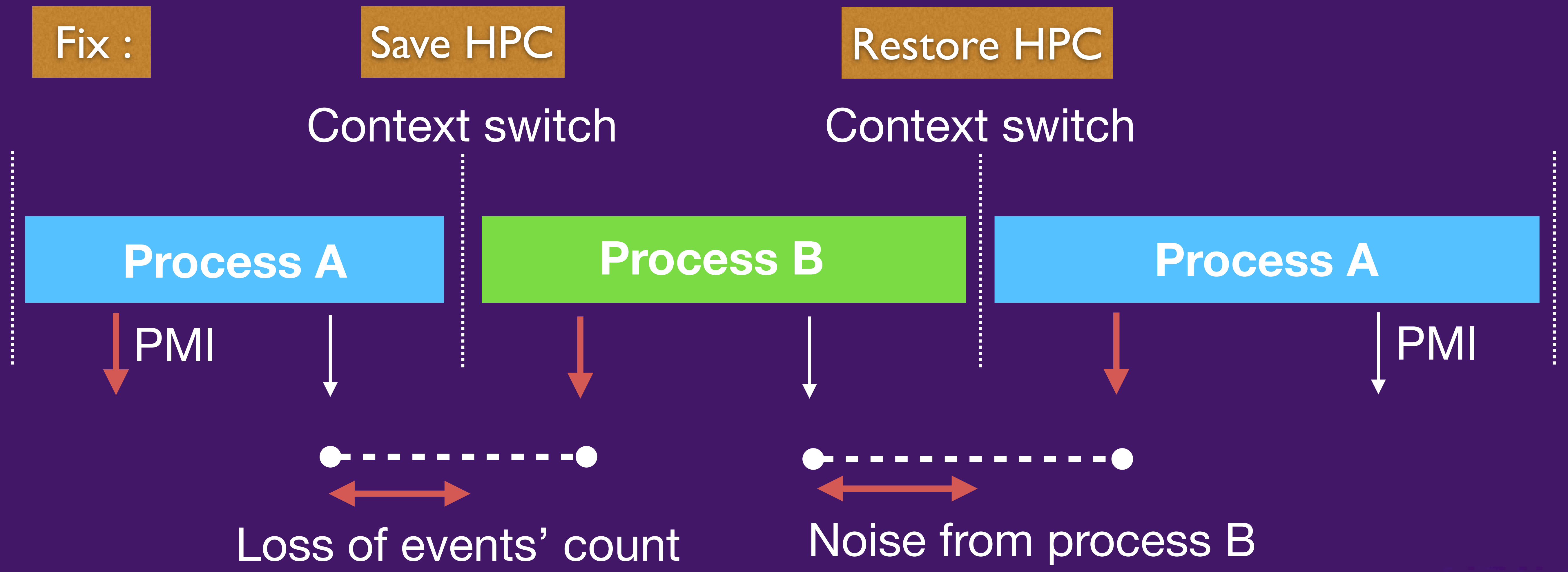


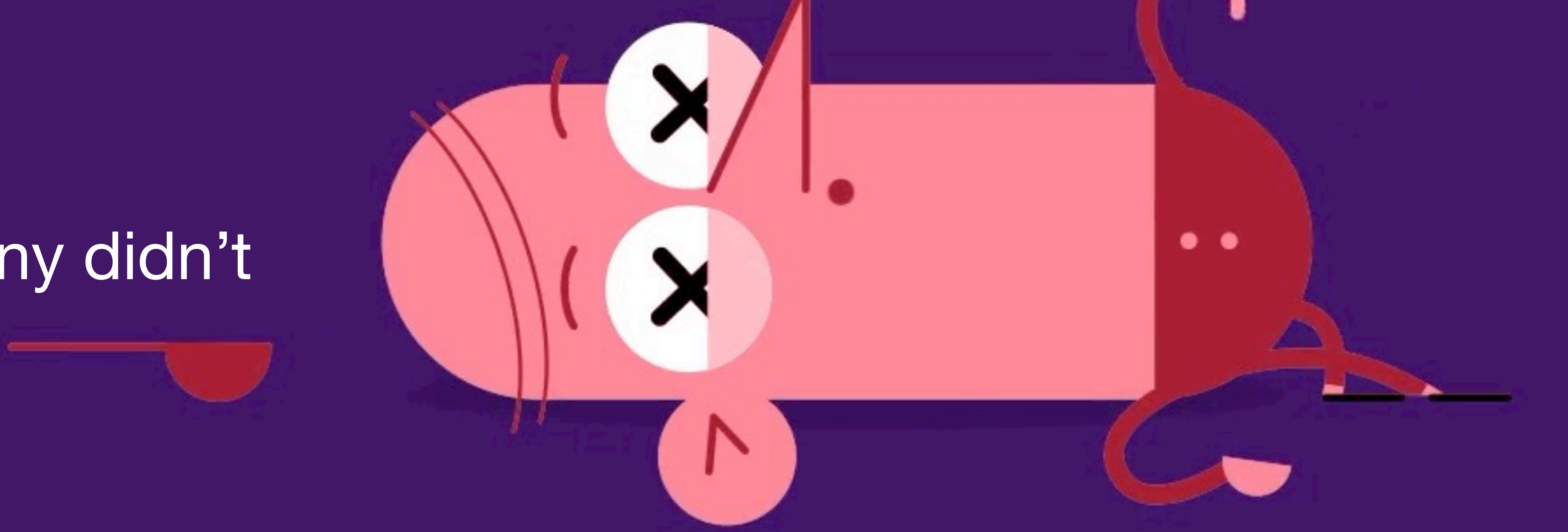
3. PMI is generated
4. At interrupt, read counter values



Mishandling of HPC Data

Filtering of processes at performance monitoring interrupt (PMI)



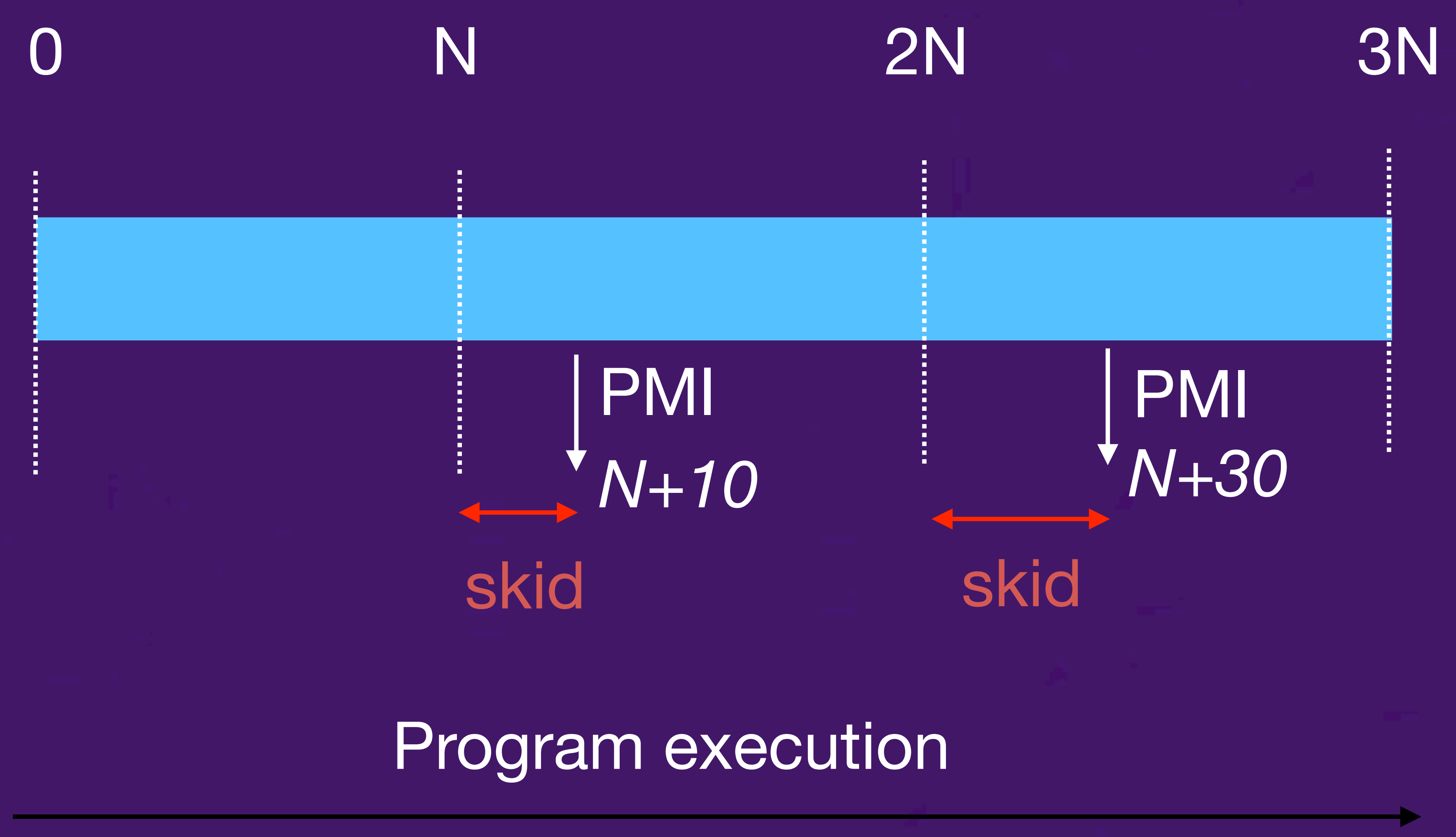


- Thankfully, there is an easy fix
  - Some papers applied this fix, but many didn't

Non-determinism: Skid

- In sampling mode:
  - Late delivery of PMI (due to skid) leads to variation in measurements
  - Fingerprint of an application may disappear (e.g., Data only attacks)

E.g., sampling every *N* DTLB misses

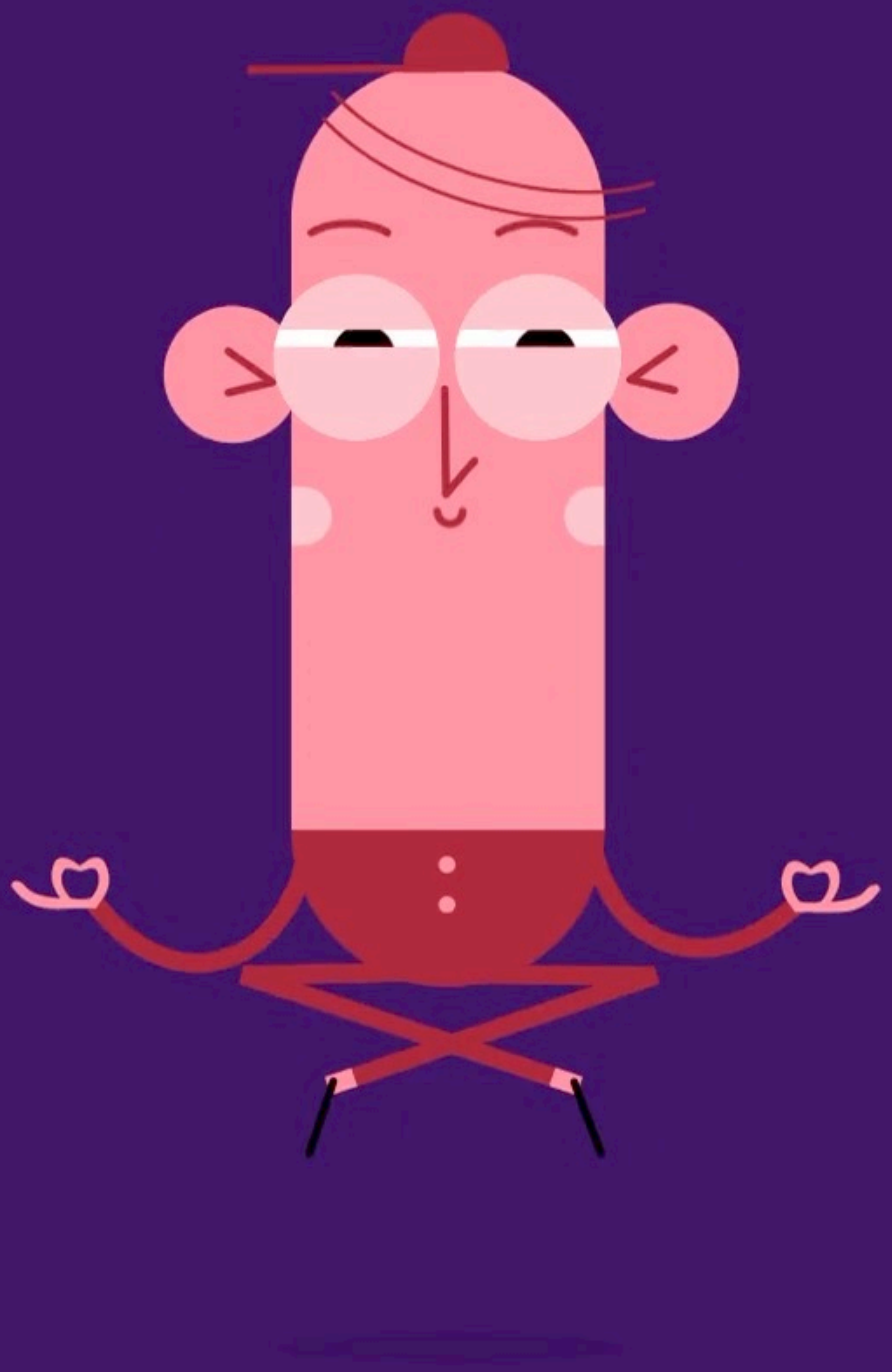


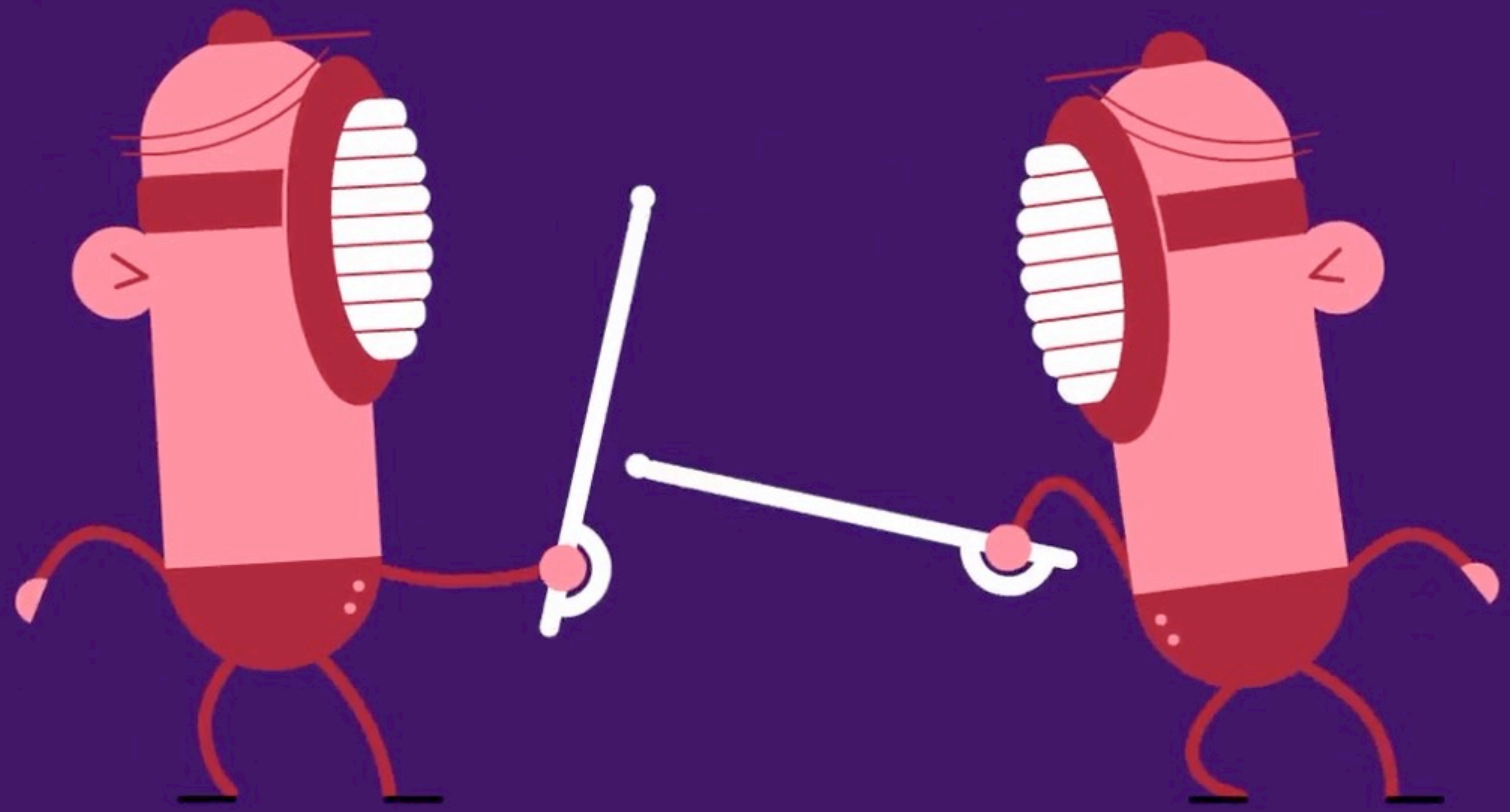
"Hardware performance monitoring for the rest of us: a position and survey" Moseley et al., Network and Parallel Computing, 2011

Non-determinism: Overcount

- We revisited the non-determinism issues based on the seminal work by Weaver & McKee [IWC, 2008]
- Several problems fixed, but some old issues persist even today
- New problem: **page faults**

Why do these issues matter from a security perspective?

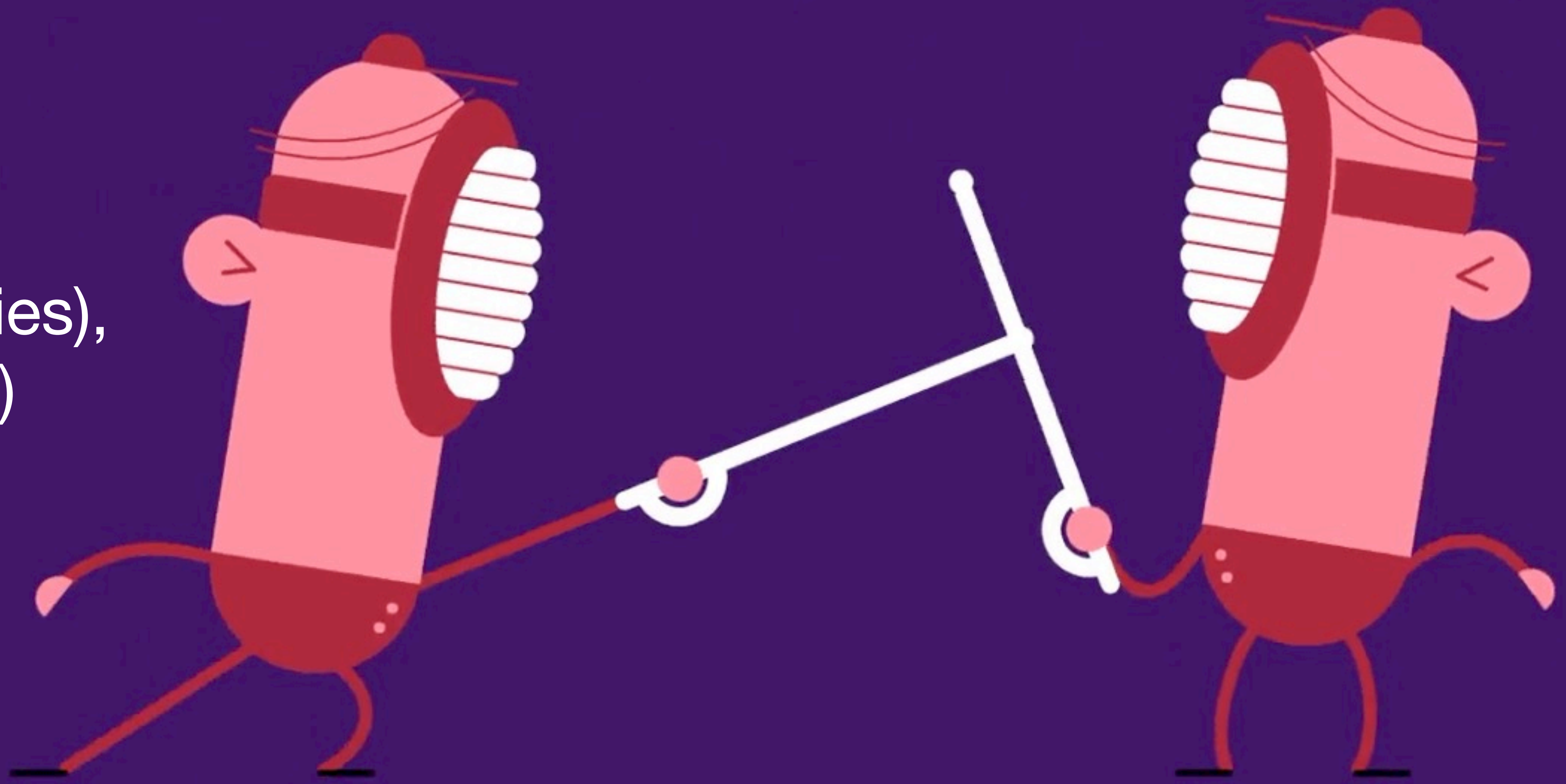




- **Improper** use of HPC in security applications can be disastrous
  - Incorrect data collection can impact the correctness of an approach
- An adversary can **manipulate** events (e.g., via page faults) to undermine defenses

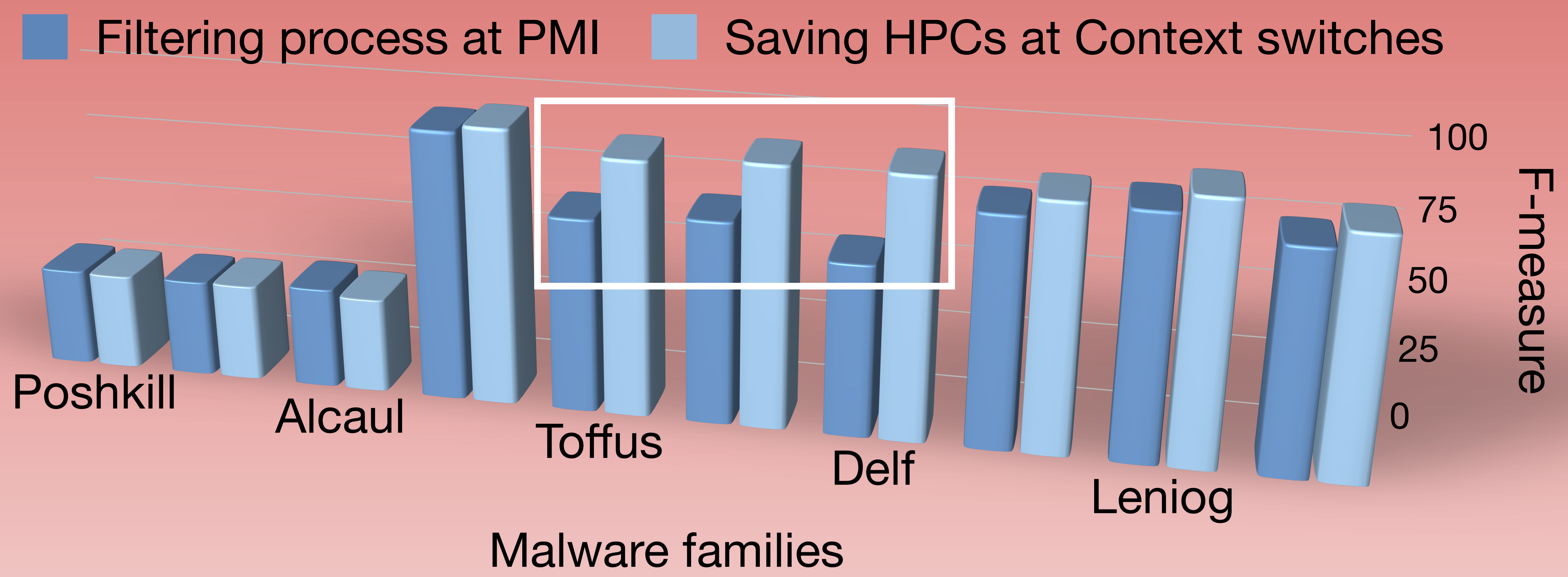
Case Study: Malware Classification

Malware (14 families),
Benign app (IE)



- Approach
  - State of the art temporal model by Tang et al. [RAID'14]
  - Sampling using PMI every *N* instructions retired
  - Events — store micro-operations, indirect call, mispredicted return and return instructions

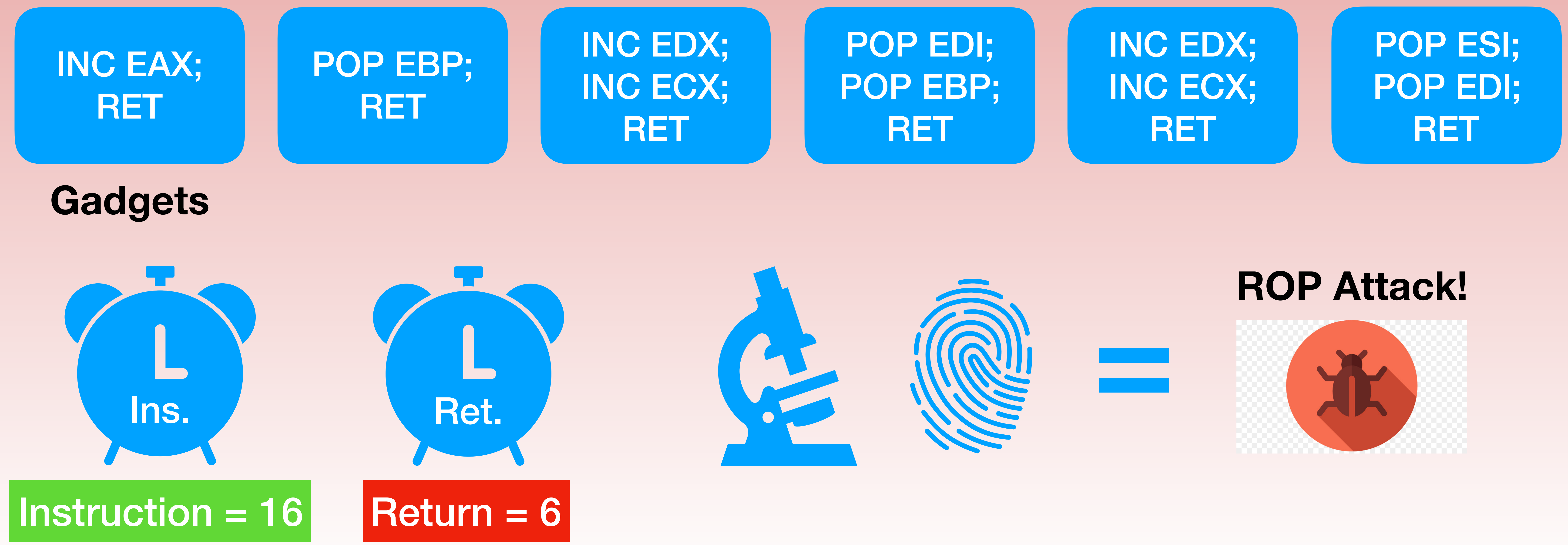
Results



- Incorrect HPC data collection significantly impacts detection accuracy
- Larger question: are HPCs a good foundation for malware detection?
    - "Hardware Performance Counters Can Detect Malware: Myth or Fact?" [Zhou et al., AsiaCCS, 2018]

Case Study: ROP Detection

- Approach
  - State of the art [Wang & Backer, arXiv, 2016]
  - For a given number of return misses, and number of instructions retired < = threshold



INC EAX; RET

POP EBP; RET

INC EDX; INC ECX; RET

POP EDI; POP EBP; RET

INC EDX; INC ECX; RET

POP ESI; POP EDI; RET

Gadgets

Instruction = 16

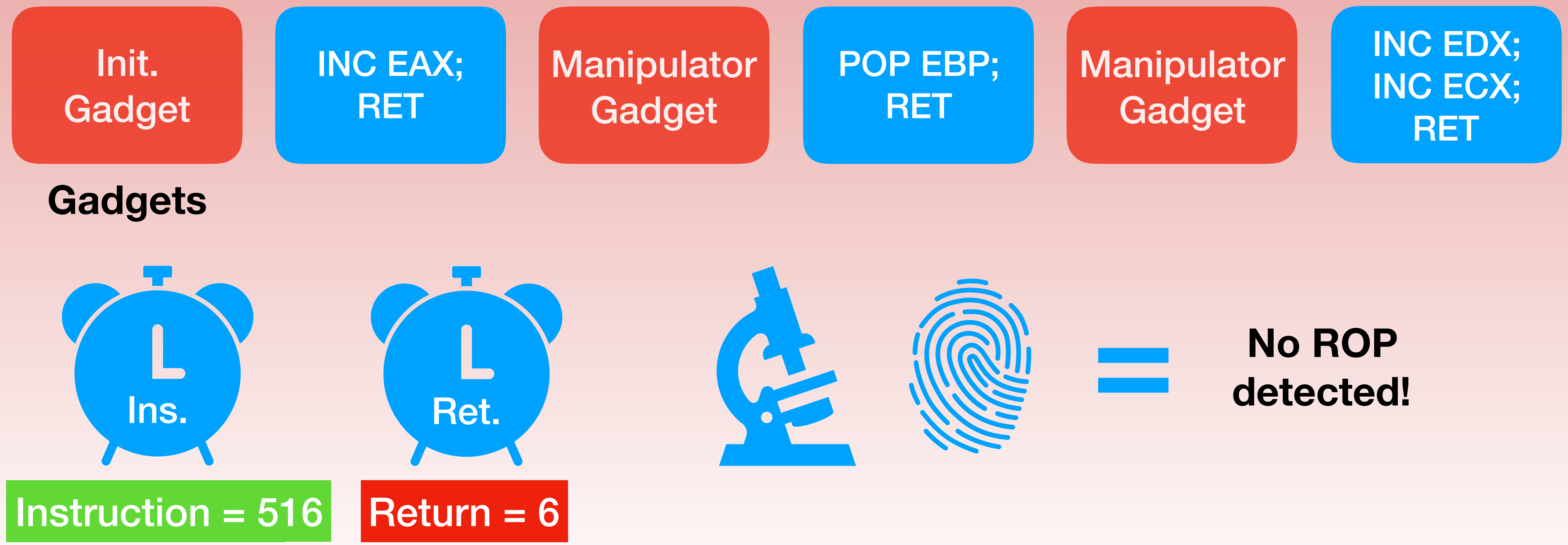
Return = 6

ROP Attack!

Case Study: ROP Detection

Results

- Irrespective of parameter choices, non-determinism can be leveraged by an adversary to bypass the ROP detection



Init. Gadget

INC EAX; RET

Manipulator Gadget

POP EBP; RET

Manipulator Gadget

INC EDX; INC ECX; RET

Gadgets

Ins.

Ret.

Instruction = 516

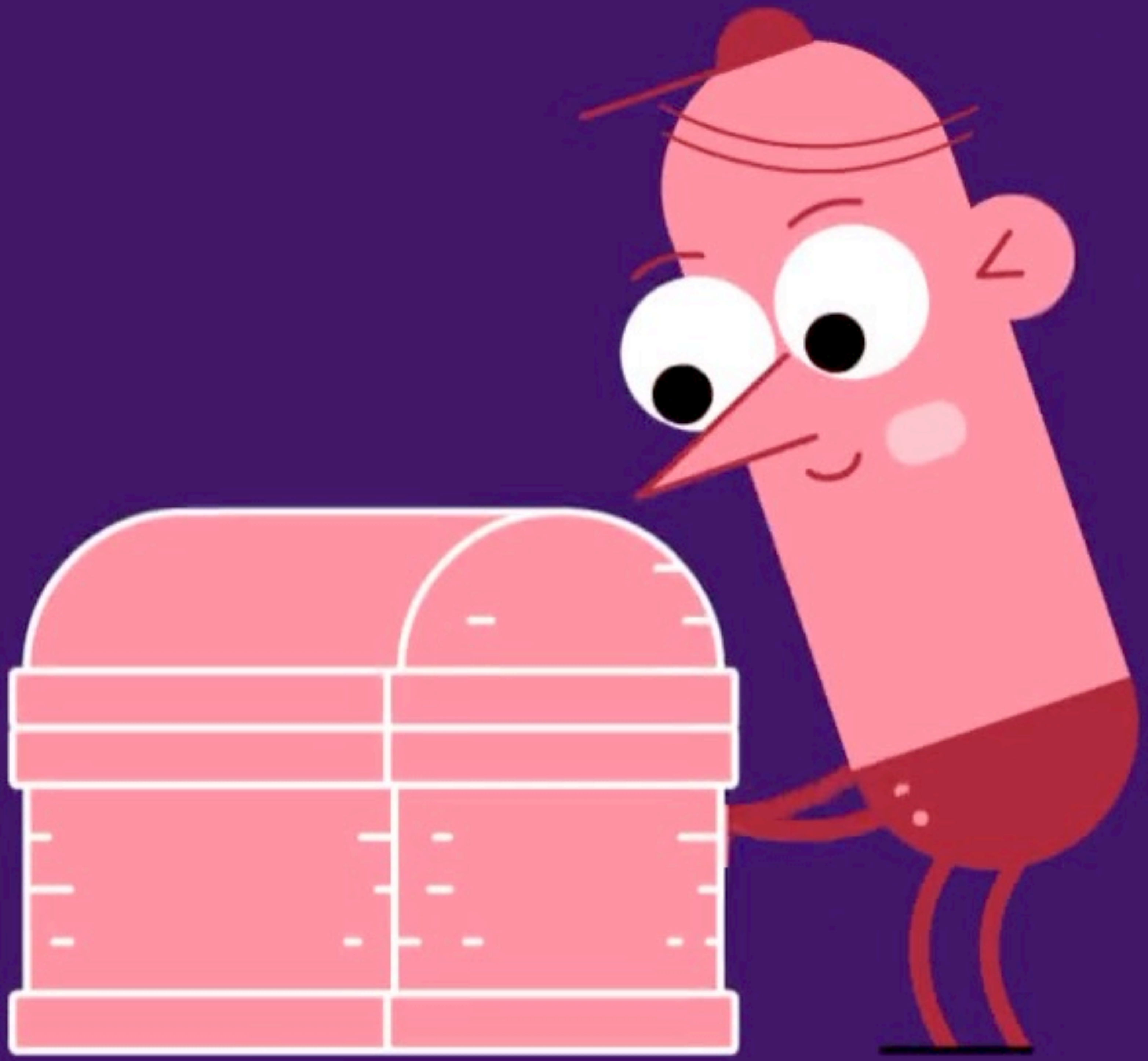
Return = 6

=

No ROP detected!

Closing remarks

HPCs offer a powerful capability, but like anything else, the devil is in the details



- We need make sure we are *not* **blindly** applying HPCs to security applications, especially defenses, in ways that go beyond their original intent
- See our recommendations on using HPCs

Questions?

sdas@cs.unc.edu