



Skills You Need To Have To Become A Published Author In Academia

Presented by:

Dr. Martin Margala

Professor of Computer Science;

Director of School of Computing and Informatics

Dr. Martin Margala is a Professor and Director of school of computer science and informatics in University of Louisiana at Lafayette, since 2021. He received his PhD degree in Electrical and Computer Engineering from the University of Alberta, Canada. He was Full Professor and Chair of the Electrical and Computer Engineering Department at the University of Massachusetts Lowell and a Co-Director of the Center for Smart Cyber-Physical Systems (SCyPS). Moreover, Dr. Margala is a senior member of ACM, IEEE in the areas of High-Performance Sustainable Computing Architectures and Design for Testability and Reliability.

In this presentation, Dr. Margala mentioned the following certain valuable points:

- Academic dissemination's categories, including conferences and journals, along with their attributes, aspects, and differences, the variations among required formats.
- The perspective of article reviewers and essential criteria they investigate in each manuscript to determine its acceptance or rejection.
- The organizational structure of appropriate articles for successful publication.
- Approach and strategies for avoiding plagiarism and how to achieve them.
- The significance of self-representation, self-identity, and presenting the paper and individuals at conferences to build a professional network for a future career.

These meetings will continue on Fridays for the next two weeks until the end of October 2024, with additional information about the dissemination of the paper.

Timestamp	First Name	Last Name	UL ID	Email ID	Are you an IEEE Member?
10/11/2024 10:14:35	Ashok	Polavarapu	C00539981	ashok.polavarapu1@louisiana.e	Yes
10/11/2024 11:13:22	Leila	Gheisi	C00497500	Leila.gheisi1@louisiana.e	Yes
10/11/2024 11:13:39	Subigya	Gautam	C00540115	gautamsubigya@gmail.c	No
10/11/2024 11:14:00	Austin	Bryant	C00481025	austin.bryant1	Yes
10/11/2024 11:14:02	Mohammad Masudur	Rahman	C00553517	C00553517@louisiana.e	Yes
10/11/2024 11:14:49	Borun	Das	C00521897	borun.das1@louisiana.ec	Yes
10/11/2024 11:14:59	Ajay	Banstola	C00565795	Ajay.banstola1@louisian	Yes
10/11/2024 11:19:32	Ashim	Sharma	C00553328	c00553328@louisiana.ec	Yes
10/11/2024 11:19:59	Krishna	Rauniyar	C00551157	C00551157@louisiana.ec	Yes
10/11/2024 11:25:03	Nabin	Pakka	C00540403	Nabin.pakka1@louisiana	No
10/11/2024 11:57:52	Arman Riaz	Ochi	C00527858	arman-riaz.ochi1.louisiar	Yes
10/11/2024 11:59:46	Md Shafiur Raihan	Shafi	C00551559	c00551559@louisiana.ec	No
10/11/2024 12:10:49	Shreeya	Pandey	C00568778	C00568778@louisiana.ec	Yes
10/11/2024 12:11:02	Roberto	Salazar	C00416436	C00416436@louisiana.ec	Yes
10/11/2024 12:12:47	Harsha Vardhan Rao	Vemuganti	C00567595	c00567595@louisiana.ec	Yes
10/11/2024 12:28:02	Niloorfar	Heidarikohol	C00476097	C00476906@louisiana.ec	Yes
10/11/2024 16:14:03	Yalda	Gheisi	C00497401	C00497401@louisiana.ec	Yes
10/11/2024 22:53:02	Jesse	Marin	C00461326	jesse.marin1@louisiana.e	No
10/14/2024 12:40:30	Niloorfar	Kolahchi	C00528374	niloorfar.kolahchi1@louis	Yes
10/14/2024 18:14:40	Xingli	Zhang	C00536580	xingli.zhang@louisiana.e	Yes
10/17/2024 19:23:33	Abu	Sayeed	C00575490	abu.sayeed1@louisiana.ec	No





Phantom: Exploiting Decoder-detectable Mispredictions

Johannes Wikner*
kwikner@ethz.ch
ETH Zürich
Zurich, Switzerland

Daniel Trujillo*
dtrujillo@ethz.ch
ETH Zürich
Zurich, Switzerland

Kaveh Razavi
kaveh@ethz.ch
ETH Zürich
Zurich, Switzerland

ABSTRACT

Violating the Von Neumann sequential processing principle at the microarchitectural level is commonplace to reach high performing CPU hardware — violations are safe as long as software executes correctly at the architectural interface. Speculative execution attacks exploit these violations and spoof up secret-dependent memory accesses allowed by long speculation windows due to the late detection of these violations in the pipeline. In this paper, we show that recent AMD and Intel CPUs speculate very early in their pipeline, even before they decode the current instruction. This mechanism enables new sources of speculation to be triggered from almost any instruction, enabling a new class of attacks that we refer to as PHANTOM. Unlike Spectre, PHANTOM speculation windows are short since the violations are detected early. Nevertheless, PHANTOM allows for transient fetch and transient decode on all recent x86-based microarchitectures, and transient execution on AMD Zen 1 and 2. We build a number of exploits using these new PHANTOM primitives and discuss why mitigating them is difficult in practice.

CCS CONCEPTS

• Security and privacy → Side-channel analysis and countermeasures; Hardware reverse engineering; Information flow control

KEYWORDS

to arbitrary information disclosure in many scenarios of interest [34, 37, 44, 8, 72, 61, 10, 20, 73]. While it is commonly assumed that the CPU speculates *only* after it decodes the instruction to be a branch, we show in this paper that all recent AMD and Intel CPUs speculate at much earlier stages of their pipeline. Our investigation into this speculation *before instruction decode* uncovers a new class of attacks that we refer to as PHANTOM speculation. We show the practical importance of PHANTOM speculation by building a number of exploits for the AMD Zen microarchitectures.

Speculation before instruction decode. In the first stage of a pipelined CPU architecture, the Instruction Fetch unit fetches blocks of instructions from the instruction cache. Instruction prefetchers try to predict future instruction cache lines and bring them into the cache before execution reaches those cache lines [77]. These predictions are made by learning the control flow of the instructions over time, and are not based on the instructions themselves. Depending on the control flow of the program, instructions that are prefetched may never enter the pipeline. We show in this paper that modern AMD and Intel CPUs do much more to improve performance: they predict and fetch the next block of instructions from the instruction cache into the pipeline immediately after the current fetch, *before* branch sources are decoded, in line with designs previously discussed in the microarchitecture community [14, 7].

The decision of whether the current instruction is a branch is made