Poster: Pathfinder: High-Resolution Control-Flow Attacks Exploiting the Conditional Branch Predictor

Hosein Yavarzadeh[†], Archit Agarwal[†], Max Christman[§], Christina Garman[*], Daniel Genkin[‡], Andrew Kwong[§], Daniel Moghimi[$], Deian Stefan[†], Kazem Taram[*], Dean Tullsen[†]

[†]University of California San Diego, [§]University of North Carolina at Chapel Hill, [*]Purdue University, [‡]Georgia Institute of Technology, [$]Google

I. Publication Information

Our research has been published in the proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS) 2024 [7]. Several media outlets have also covered our findings, including *The Hacker News* [2], *HPC Wire* [4], and *Cyber Security News* [3].

II. Introduction

Microarchitectural side-channel attacks exploit processor architectural state to leak information from one process or protection domain to another, when there should be no communication between them. Side-channel attacks can be used standalone to leak information, but are also often critical building blocks for more sophisticated attacks. For example, Spectre attacks [6] use side channels both to trigger a controlled misspeculation and to transfer data back to the attacker via transient microarchitectural state. Microarchitectural attacks can exploit various shared units within the processor such as caches, branch predictors, and address translation buffers.

While there have been several control-flow based side channel attacks on the branch predictor, they are hampered by limited knowledge of the associated addressing techniques. In particular, attacks targeting the conditional branch predictor (CBP) exclusively target the simplest structure in the predictor, which only enables extracting or injecting coarse control-flow information.

III. Our Contributions

In this work, we demonstrate a new class of control flow attacks which exploit detailed knowledge of every aspect of modern CBPs [8]. We show that due to recent research that fully exposed the internal structure of modern Intel branch predictors, prior limitations no longer exist, making conditional branch predictors a powerful and dangerous medium for attack. We create primitives that make it as easy (from a programmer's perspective) to read from and write to the tables of the conditional branch predictor, or the path history register (PHR, a precise record of the last taken branches), as it is to read and write memory. No prior work has used the PHR as an attack vector. We show that the ability to read and write the PHR precisely is a particularly powerful primitive that enables two new attack capabilities.

A. Full Control-Flow Recovery

Reading the path history register (PHR) provides several advantages over previous side-channel attacks exploiting branch predictors and caches, as it records the complete control flow history of recent branches including branch addresses and precise ordering. We can target each individual execution of a branch that is executed many times, where prior CBP attacks could only influence the first few, or capture the bias of the last few instances. We can detect the outcome of every instance of each branch directly, regardless of the cache footprint of the executed code or even whether any memory data was touched.

In this paper, we describe the key primitives that allow us to read and write the PHR, and to read and write the prediction history tables (PHTs) of the CBP. We further describe more advanced techniques. The first of those allows us to go beyond just the recovered PHR and capture control flow history of nearly unlimited length, allowing us to track extremely large swaths of branch history. Previous attacks that can precisely track the entire program's control flow require elevated privilege (i.e., root access), which makes them practical against the specific threat model of untrusted OS (e.g., against Intel SGX), but not valid for other threat models.

Finally, we also describe a tool that transforms the PHR (a series of heavily folded bits of branch address and target history) into a (nearly always) unambiguous control flow graph (CFG) which includes the full history (series of taken/not taken decisions) of every branch. This can be done both for the physical PHR (194 taken branches and all intervening not-taken branches), but also for our extended path history. Please refer to the full paper [7] for more details.

B. High-Resolution Spectre Attack

Beyond just capturing a complete control flow history, writing to the PHR gives us the ability to launch extremely high resolution poisoning (e.g., Spectre) attacks. Prior Spectre-style attacks typically only influence the first instance (or the first few, all in the same direction) of a branch at a given address, such as boundary checking 'if' statements. However, as mentioned, with complete control over the PHR and the

PHTs, we can now influence an arbitrary instance of a branch executed many times, inducing sophisticated patterns of branch mispredictions, for example, at specific iterations of a 'for' loop. For example, we can cause an iterative encryption or decryption algorithm to complete and return after any number of iterations (both more and less than intended). This allows us, for example, to observe the results of every incremental step in the algorithm, which also reveals the key.

We demonstrate the implications of these attack primitives with two case studies: (1) We demonstrate a speculative execution attack against AES that returns intermediate values at multiple steps to recover the AES key, which requires both the reading and writing primitives. (2) We also steal secret images by capturing the complete control flow (up to tens of thousands of branches) of *libjpeg* routines. Finally, we demonstrate that our attack primitives, and thus also the attacks built upon them, work across virtually all protection boundaries and in the presence of all recent control-flow mitigations from Intel.

IV. Responsible Disclosure

We communicated the security findings outlined in the paper to both Intel and AMD in November 2023. Intel has informed other affected hardware/software vendors about the issues.

Intel has shared their plans to address the concerns raised in the paper through a Security Announcement, INTEL-2024-04-26-001-Pathfinder [5]. AMD addressed the concerns raised in the paper through a Security Bulletin, AMD-SB-7015 [1].

V. Attack Results

A. Image Recovery Attack: libjpeg

We use the *Read PHR* attack primitive in conjunction with Pathfinder to uncover the precise runtime control flow of the image decompression process, particularly within the IDCT function. This enables us to identify which rows/columns are constant within each block. Additionally, we know that the number of constant rows/columns in a block corresponds to the block's relative complexity. We use this insight to recover the original image, as we assign each 8 8 block a value based on the normalized number of constant rows/columns within it. Consequently, the recovered image frequently exhibits a high similarity to the results of edge detection, particularly in blocks positioned along the image's edges, which tend to be more complex than others.

We conducted an evaluation using a test set of 15 JPEG images. To create a diverse collection, we included a range of images, including high-resolution photographs, simpler logo-style images, QR codes, captchas, and more. Figure 1 presents three examples of successful image recovery using our attack method. The first example demonstrates the recovery of a QR code from the ASPLOS website, which remains scannable despite some noise. The second and third examples showcase the recovery of the ASPLOS logo and website background image, respectively.

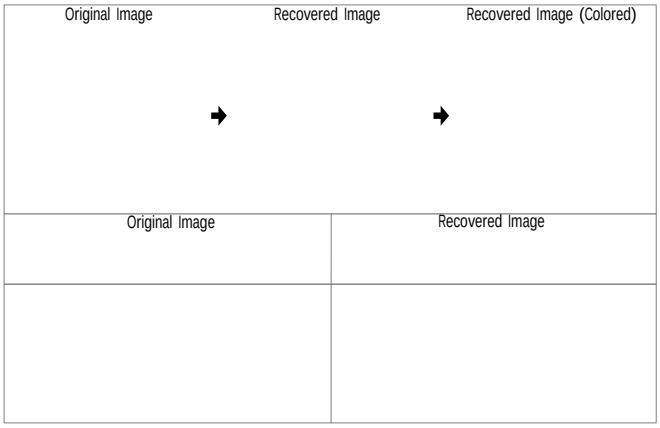

Fig. 1. Examples of Recovered Images by *Pathfinder*.

B. Leaking AES Keys

We developed a new Spectre-style attack on the widely used AES algorithm to extract *encryption keys*. We show that reduced-round ciphertexts can be captured via speculative execution channels, and used in conjunction with the full-round ciphertext to recover the AES key, even from constant-time implementations that leverage AES-NI hardware extension.

We use the Intel-IPP AES ECB Encryption function as our victim model, utilizing a 128-bit key for encrypting a 128-byte data block. As previously mentioned, our attack is capable of speculatively terminating the victim loop at any iteration, in this case ranging from the first to one less than the total number of rounds. We rigorously test all of these, leaking the ciphertext and subsequently verifying the correctness of the recovered reduced-round ciphertext.

References

[1] Exploiting the conditional branch predictor. https://www.amd.com/en/resources/product-security/bulletin/amd-sb-7015.html, 2024.

[2] New spectre-style 'pathfinder' attack targets intel cpu, leak encryption keys and data. https://thehackernews.com/2024/05/new-spectre-style-pathfinder-attack.html, 2024.

[3] Pathfinder - new attack steals sensitive data from modern processors. https://cybersecuritynews.com/pathfinder-processor-data-breach/, 2024.

[4] Processor security: Taking the wong path. https://www.hpcwire.com/2024/05/09/processor-security-taking-the-wong-path/, 2024.

[5] Security announcement: Intel-2024-04-26-001-pathfinder. https://www.intel.com/content/www/us/en/security-center/announcement/intel-security-announcement-2024-04-26-001.html, 2024.

[6] Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, et al. Spectre attacks: Exploiting speculative execution. *Communications of the ACM*, 63(7):93–101, 2020.

[7] Hosein Yavarzadeh, Archit Agarwal, Max Christman, Christina Garman, Daniel Genkin, Andrew Kwong, Daniel Moghimi, Deian Stefan, Kazem Taram, and Dean Tullsen. Pathfinder: High-resolution control-flow attacks exploiting the conditional branch predictor. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS '24, page 770–784, 2024.

[8] Hosein Yavarzadeh, Mohammadkazem Taram, Shravan Narayan, Deian Stefan, and Dean Tullsen. Half&half: Demystifying intel's directional branch predictors for fast, secure partitioned execution. In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 1220–1237. IEEE Computer Society, 2023.