Nicholas Hunter Mosier

PhD Candidate at Stanford University, Department of Computer Science

47 Olmsted Rd, Apt 222 • Stanford, CA • 94305

Phone: (520) 250-2480 • Email: nmosier@stanford.edu

LinkedIn: in/nmosier • GitHub: nmosier • Website: https://nmosier.github.io

EDUCATION

Stanford University • Stanford, CA *September 2020 – present*

PhD Candidate in Computer Science at Stanford University, School of Engineering

- GPA: 4.23 / 4
- Advisor: Caroline Trippel
- Expected graduation / conferral of PhD: March 2026

Middlebury College • Middlebury, VT *September 2016 – May 2020*

Bachelor of Arts with a Double Major in Computer Science and Mathematics

- GPA: 3.99 / 4
- Advisor: Peter C. Johnson

PHD RESEARCH SUMMARY

Security-critical software—including OS kernels, cryptographic libraries, and web browsers—is written under the assumption that it executes sequentially. However, in reality, it does not. Modern processors execute programs speculatively and out-of-order, enabling *transient execution*—i.e., the execution of instructions that are never architecturally committed. Attacks such as Spectre and Meltdown exploit transient execution to steer secret data towards the unsafe operands of transient transmit instructions, which leak data via microarchitectural side channels like the cache. My research focuses on comprehensively and provably securing security-critical software against transient execution attacks using formally-grounded compiler-based techniques (e.g., Serberus, implemented for LLVM) and hardware-software codesigned mitigations (e.g., TPT-PTeX, implemented for LLVM and gem5).

AWARDS & HONORS

• 1st place team in Microarchitectural Attacks and Defenses (MAD) CTF at ISCA'23	Jun. 2023
• Graduated Salutatorian of the Class of 2020 at Middlebury College	May 2020
• Graduated Summa Cum Laude from Middlebury College in 2020	May 2020
• Timothy T. Huang Award in Computer Science, Middlebury College	May 2020
• Phi Beta Kappa Prize, Middlebury College	Nov. 2019
• Induction into Phi Beta Kappa, Middlebury College	Nov. 2019

PUBLICATIONS

Refereed

- **Nicholas Mosier**, Hamed Nemati, John C. Mitchell, Caroline Trippel. "Serberus: Protecting Cryptographic Code from Spectres at Compile Time." In Proceedings of the 45th IEEE Symposium on Security and Privacy (S&P'24). San Fransisco, California. May 2024.
- Sonia Martin, **Nicholas Mosier**, Obi Nnorom Jr., Yancheng Ou, Liana Patel, Oskar Triebe, Gustavo Cezar, Philip Levis, Ram Rajagopal. "Software Defined Energy Grid Storage." In Proceedings of the 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys '22), Boston, Massachusetts. November 2022.
- **Nicholas Mosier**, Hanna Lachnitt, Hamed Nemati, Caroline Trippel. "Axiomatic Hardware-Software Contracts for Security." In Proceedings of the 49th Annual International Symposium on Computer Architecture (ISCA'22), New York City, New York. June 2022.

Non-Refereed / Non-Proceedings

- **Nicholas Mosier**, Hamed Nemati, John C. Mitchell, Caroline Trippel. "Analyzing and Exploiting Branch Mispredictions in Microcode." arXiv preprint arXiv:2501.12890v1. January 2025.
- **Nicholas Mosier**, Kate Eselius, Hamed Nemati, John C. Mitchell, Caroline Trippel. "Hardware-Software Codesign for Mitigating Spectre." In the Workshop on Programming Languages for Architecture (PLARCH'23) at the 50th International Symposium on Computer Architecture. June, 2023.

In Preparation

- **Nicholas Mosier**, Hamed Nemati, John C. Mitchell, Caroline Trippel. "Taint Primitive Tracking: Transient Hardware Enforcement of Software-Defined Architectural Protection Policies." In preparation; planned submission March 2025.

RESEARCH EXPERIENCE

μSpectre: a New Class of Transient Execution Attacks • Stanford University *February 2024 – present*

Research project with Hamed Nemati, and John C. Mitchell, and Caroline Trippel (PI)

- Discovered *μ*Spectre, a new class of transient execution attacks that exploit microcode branch mispredictions
- Identified that two previously discovered attacks, Rogue System Register Read and Zero Dividend Injection, are in fact instances of *μ*Spectre
- Identified a novel *μ*Spectre vulnerability in Intel Goldmont microcode that allows an attacker to transiently read from an architecturally-inaccessible microarchitectural buffer
- Proposed *μ*SLH, a microcode-based mitigation for *μ*Spectre attacks
- *Ongoing work:* Developing a tool for automatic black-box detection of vulnerable microcode branches on Intel and AMD processors
- Prepublished: arXiv:2501.12890

Applying Taint Primitive Elimination to Industry Simulators • Intel Corporation *June 2024 – August 2024*

Research project with Scott Constable (mentor) and Carlos Rozas (manager)

- Implemented Taint Primitive Elimination, a simplified variant of Taint Primitive Tracking that I developed at Stanford, on an industrial in-house simulator to gauge its performance impact
- Internship was a success, but cannot share more details due to non-disclosure agreements

Taint Primitive Tracking and Protection Type Extensions • Stanford University *Fall 2022 – present*

Research project with Hamed Nemati, John C. Mitchell, and Caroline Trippel (PI)

- Developed TPT-PTeX, the most performant Spectre defense to enforce speculative non-interference to date
- Designed defense as two hardware-software codesigned ISA/hardware extensions: Protection Type Extensions (PTeX) to allow software to communicate to hardware what data requires protection, and Taint Primitive Tracking (TPT) to provably enforce in hardware that no protected data leaks transiently using our theory of taint primitives
- Engaged in extensive hardware-software codesign to minimize hardware complexity by offloading program leakage analysis to software via PTeX ISA extensions
- Implemented program leakage analysis as a LLVM x86 machine IR pass and modeled PTeX and TPT hardware extensions in gem5 O3 processor simulator
- Evaluated TPT-PTeX on the SPEC CPU2017 benchmarks and observed 18%/45% runtime overhead on an Intel Alder Lake E-core/P-core configuration, compared to the state of the art defense SPT with 30%/67% overhead

- Developed Cross-Binary SimPoints, a new methodology for evaluating hardware-software codesigned defenses in gem5 that overcomes the inaccuracies and inefficiencies of traditional SimPoint-based techniques in this context
- Preparing to submit to TPT-PTeX to top-tier 2026 conference in March 2025

Comprehensive Compile-Time Spectre Mitigations • Stanford University • (GitHub) *Spring 2022 – present*

Research Project with Hamed Nemati, John C. Mitchell, Caroline Trippel (PI)

- Developed Serberus, the first comprehensive software-based Spectre mitigation for constant-time cryptographic code
- Implemented Serberus as three programmer-transparent, intraprocedural LLVM passes, requiring no secrecy labels or programmer annotations
- Used our novel theory of taint primitives to proved the completeness of Serberus' security guarantees
- Evaluated Serberus on crypto benchmarks, on which it averages 21.5% overhead, outperforming state-of-the-art Spectre mitigations despite offering stronger security guarantees
- Inspired Intel to codify the precise speculative semantics of Intel CET-IBT (used by Serberus) for existing processor implementations in the Intel Software Developer Manual
- Published at S&P'24
- *Ongoing work:* Adapting Serberus to leverage Intel MPK to allow the protection of only secret-processing functions (rather than all code), hypothesized to drastically lower performance overheads

Leakage Containment Models • Stanford University • (GitHub) *Summer 2021 – Summer 2022*

Research project with Hamed Nemati, John C. Mitchell, and Caroline Trippel (PI)

- Helped to develop leakage containment models, an axiomatic contract for modeling microarchitectural leakage
- Designed and implemented Clou, a LLVM plugin for detecting Spectre-PHT and -STL leakage in C programs at compile time
- Analyzed the widely-used libsodium and OpenSSL crypto libraries for Spectre vulnerabilities
- Discovered and reported multiple libsodium and OpenSSL Spectre vulnerabilities, covered in OpenSSL blog post
- Clou's findings inspired my discovery *taint primitives*, which serve as the formal foundation for all of my subsequent Spectre defense projects
- Published at ISCA'22

PRESS

Spectre and Meltdown Attacks Against OpenSSL • OpenSSL Blog • (link) *May 2022*

- Discusses Spectre vulnerabilities discovered by our Spectre vulnerability detection tool, Clou

TALKS & PRESENTATIONS

Title	Venue	*Date*
Protection Type Extensions	Intel Security & Privacy Research Intern Report-Out	*Aug. 2024*
Serberus: Protecting Cryptographic Code from Spectres at Compile Time (link)	IEEE Symposium on Security and Privacy (S&P'24)	*May 2024*
Hardware-Software Codesign for Efficiently Mitigating Transient Execution Attacks	Intel Labs Security and Privacy Research Tech Talks	*Feb. 2024*
Serberus: Protecting Cryptographic Code from Spectres at Compile Time	Intel Scalable Assurance Annual Workshop	*Sep. 2023*
Hardware-Software Codesign for Mitigating Spectre (link)	Programming Languages for Architecture (PLARCH'23, link)	*Jun 2023*

Comprehensively Mitigating Transient Execution Attacks (link)	Stanford Computer Forum Annual Meeting: Security Workshop (link)	*Apr 2023*
Axiomatic Hardware-Software Contracts for Security (link)	49th International Symposium for Computer Architecture (ISCA'22)	*Jun 2022*
Axiomatic Hardware-Software Contracts for Security (link)	Stanford Computer Forum Annual Meeting: Security Workshop (link)	*Apr 2022*
Bypassing ASLR with Speculative Buffer Overflows	Middlebury College Computer Science Undergraduate Thesis Presentation	*May 2020*
ROP with a 2nd Stack, or This Exploit is a Recursive Fibonacci Sequence Generator (link)	BSides Las Vegas (link)	*Aug. 2019*

RELEVANT COURSEWORK

- **Stanford University**: Formal Methods for Computer Systems, Computer Systems Architecture, Computer and Network Security, Program Analysis & Optimization, Artificial Intelligence, Programming Languages
- **Middlebury College**: Systems Security, Compiler Design, Operating Systems, Embedded Systems, Systems Programming, Computer Architecture, Parallel Computing, Algorithms & Complexity, Graph Theory, Combinatorics

SKILLS AND INTERESTS

Technical Skills

- Programming languages: C, C++, x86, bash, Python, Swift, Java, SystemVerilog, ARM, RISC-V, z80
- UNIX-based programming environments (Linux, macOS, FreeBSD, WSL)
- Libraries and projects: gem5, LLVM
- Tools: Ghidra, gdb, Docker, make, CMake, git, KVM, Snakemake

Interests: hardware and binary security, compiler design, formal methods, reverse engineering, computer architecture, bug hunting

Bug Hunting: discovered and received payout for 7 remote code execution (RCE) vulnerabilities in video game software

Open Source Contributions

- gem5: Merged 30 patches into the widely-used gem5 academic simulator, fixing 20 correctness bugs, fixing 3 performance issues, and adding 4 missing features. As a direct result of my contributions, gem5's KVM and O3 CPUs can now run the SPEC CPU2017 Integer and Floating-Point benchmarks.

Competitive Programming • Middlebury College

- Ran weekly programming contest practice sessions (Spring 2017 – Fall 2019)
- Participated in many programming contests on teams representing Middlebury College, such as the ACM-ICPC Preliminary Contest (Fall 2016 & 2017) and CCSCNE Contest (Spring 2017, 2019)