Context-Sensitive Fencing:
Securing Speculative Execution via Microcode Customization

Ashish Venkat, University of Virginia

http://www.cs.virginia.edu/venkat/

Abstract

Context-Sensitive Fencing (CSF) is a microcode-level defense against multiple variants of *Spectre*. CSF leverages the ability to dynamically alter the decoding of the instruction stream, to seamlessly inject new micro-ops, including fences, only when dynamic conditions indicate they are needed. This enables the processor to protect against the attack, but with minimal impact on the efficacy of key performance features such as speculative execution. This also examines several alternative fence implementations, and introduces three new types of fences which allow most dynamic reorderings of loads and stores, but in a way that prevents speculative accesses from changing visible cache state. These optimizations reduce the performance overhead of the defense mechanism, compared to state-of-the-art software-based fencing mechanisms by a factor of six.

Background and Introduction

Speculative Attacks:

Spectre Variant 1:

```
if ( x < array1_size)
  y = array2[array1[x] * 64];
```

Memory and Cache Status

```
array1_size = 0x00000008
Memory following array1 base address:
  ➤ 8 bytes of data
  ➤ … N bytes of other memory…
  ➤ 03 F1 98 A7 … (Secret)

array2 [ 0*64]
array2 [ 1*64]
array2 [ 2*64]
array2 [ 3*64]      } Content doesn't matter
array2 [ 4*64]
array2 [ 5*64]
        …
```

Uncached | Cached

Variant	Vulnerability Name
Spectre v1	Bounds Check Bypass (BCB)
Spectre v2	Branch Target Injection (BTI)
Spectre v3	Rogue Data Cache Load (RDCL)
Spectre v3a	Rogue System Register Read (RSRD)
Spectre v4	Speculative Store Bypass (SSB)
Spectre-NG	Lazy FP State Restore
Spectre v1.1	Bounds Check Bypass Store (BCBS)
Spectre v1.2	Read-only Protection Bypass
Spectre v5	Ret2Spec and SpecRSB
NetSpectre	Remote Bounds Check Bypass
Foreshadow	L1 Terminal Fault

Methodology

state-of-the-art defenses: constraining the order instructions by fence/ serializing instructions

```
mov eax, arr1_size
cmp edi, eax
jge END_LBL
mov eax, edi

mov eax, [eax+arr1]
shl eax, 0x8
mov eax, [eax+arr2]
mov [y], eax
END_LBL:
```

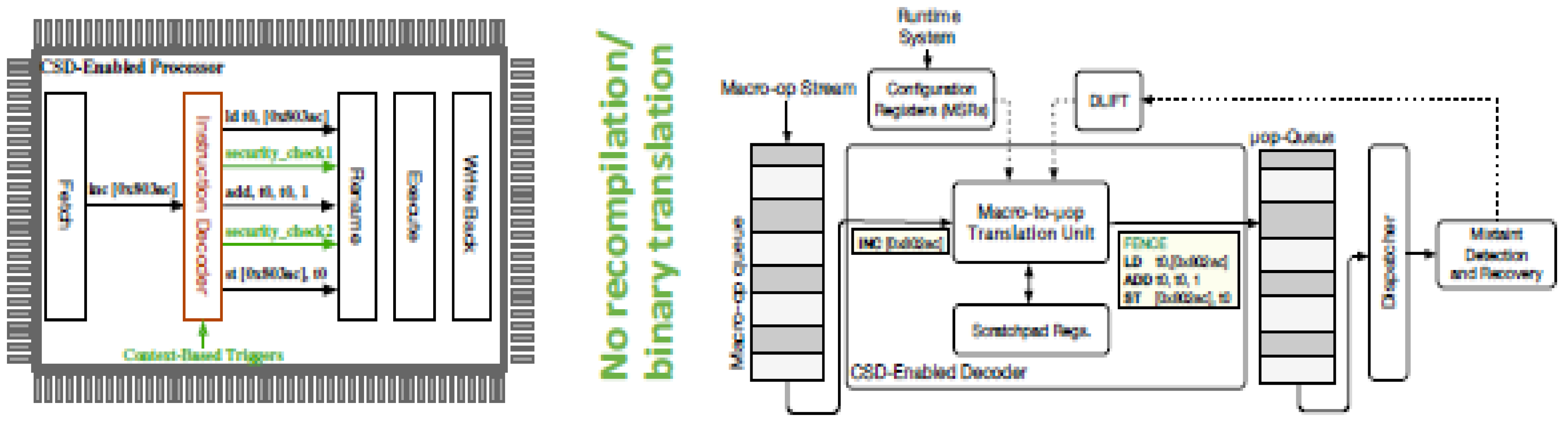
(a) Vulnerable gadget (x86)

```
mov eax, arr1_size
cmp edi, eax
jge END_LBL
mov eax, edi
___FENCE___
mov eax, [eax+arr1]
shl eax, 0x8
mov eax, [eax+arr2]
mov [y], eax
END_LBL:
```

(b) Fenced gadget (x86)

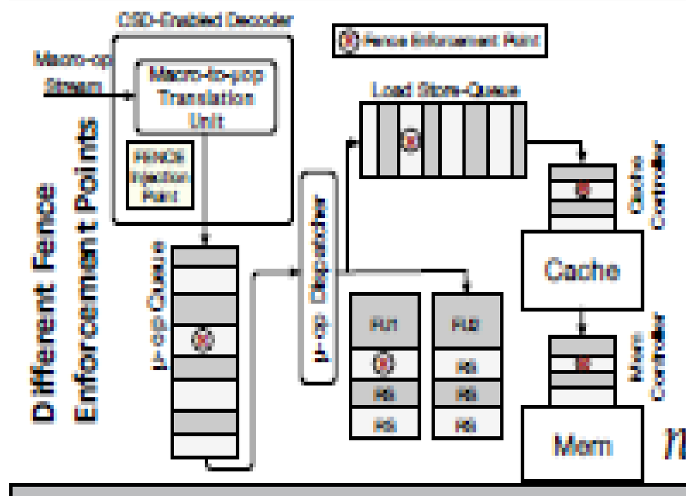
Liberal fence insertion severely hurts performance

CSD: a Micro-op Customization Framework



No recompilation/ binary translation

Newly Proposed Fences



Characteristics of Different Fence Types

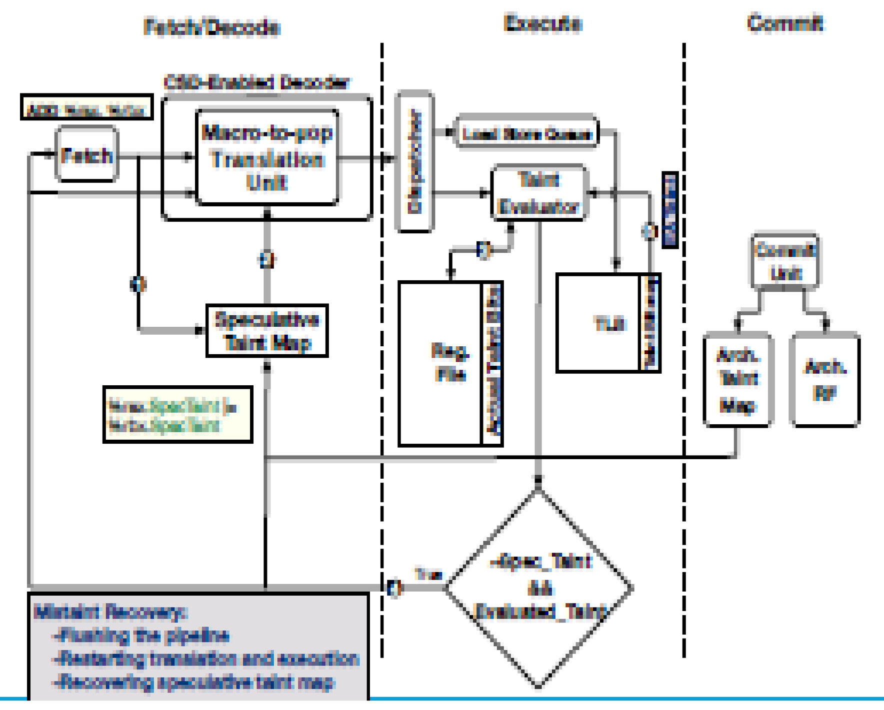
Fence Name	Enforcement Point	Strict/ Relaxed	Instructions not Allowed	Mitigates Variants	Existing/ New?
Intel's SIs (CPUID)	Fetch	Strict	All	All	Existing
LFENCE	IQ	Strict	All	All	Existing
LSQ-LFENCE	LSQ	Relaxed	Ld	v1	New
LSQ-MFENCE	LSQ	Relaxed	Ld&St	v1,v1.1,v1.2	New
CFENCE	CC*	Relaxed	None	v1	New

* CC: Cache Controller, RS: Reservation Station

non-modifying loads are allowed to pass through the *CFENCE*

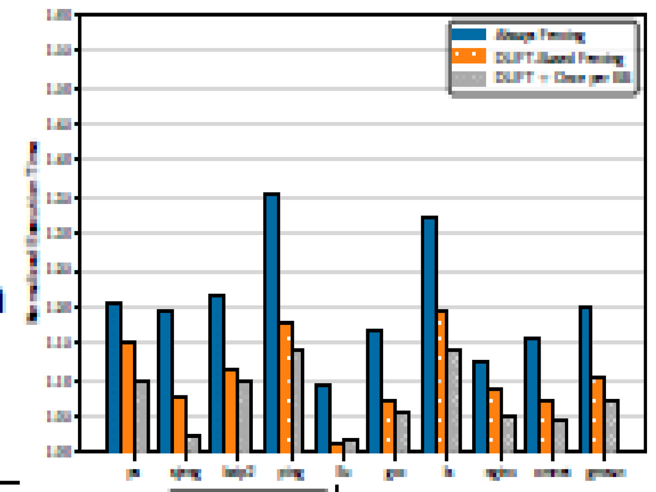
DLIFT: a Dynamic Information Flow Tracker in Spectre Era

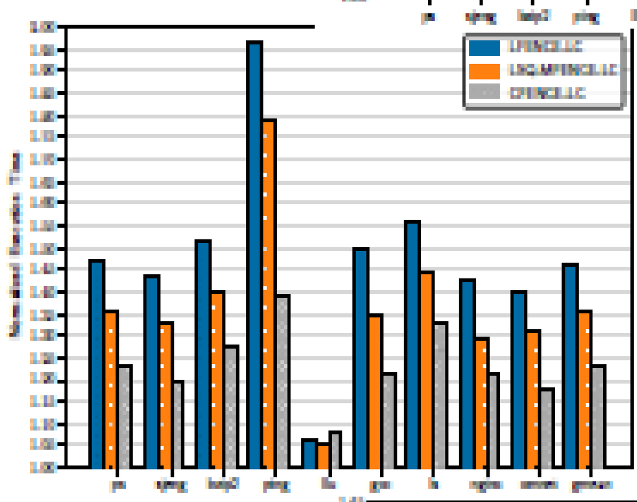
DLIFT identifies tainted accesses "*before*" the stages that enable speculative execution.



Results

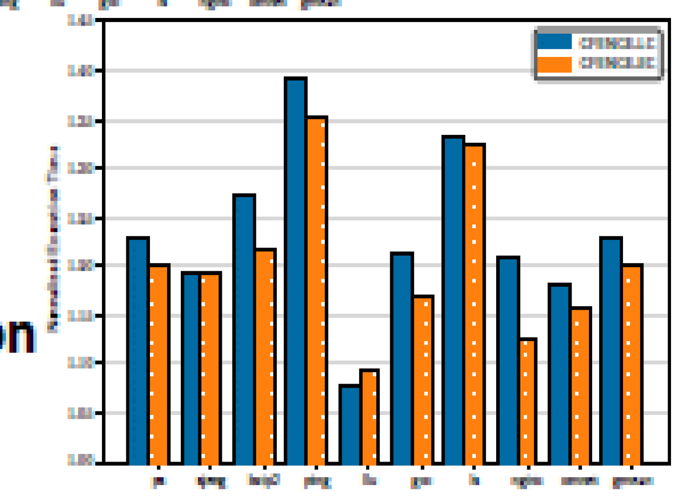
Fence Frequency Optimization



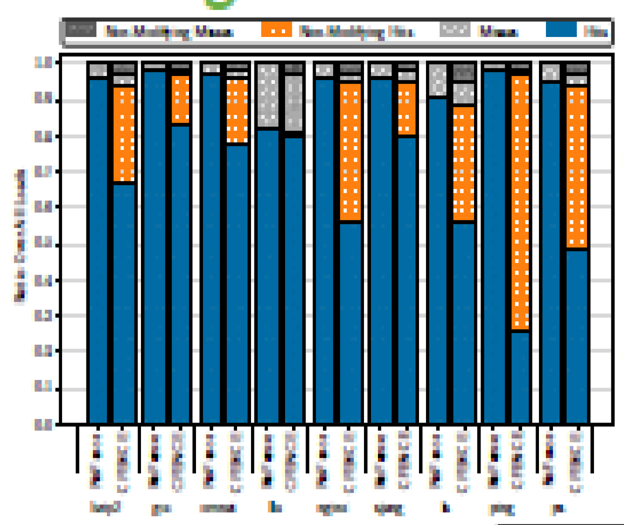


Fence Enforcement Optimization

Fence Expiration Optimization

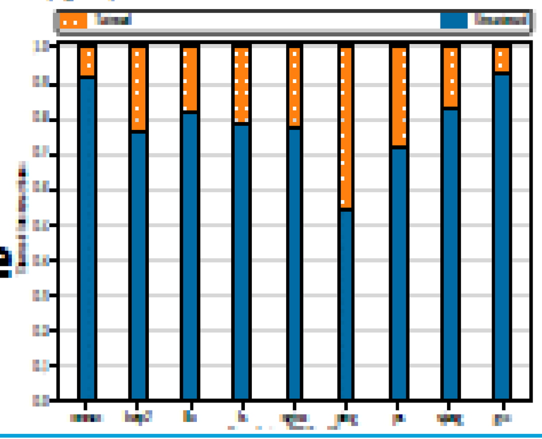


CSF reduces the cost of mitigation from 48% to < 8%.



CFence Effects on L1 Cache

DLIFT Performance



Broader Impacts:

This research tackles a highly evasive class of microarchitectural exploits that impact nearly every computer in the world, while dramatically reducing the prohibitively high performance costs of state-of-the-art software-based defenses.

The PI is extremely committed to diversity efforts within the department and in the community. In particular, the PI has been providing research mentorship to female undergraduate researchers via the ENLACE program at UC San Diego and the JUMP URI program at the University of Virginia.

The PI is supporting and providing mentorship to 2 CS graduate students for two years through this program, and has taught a new graduate project/seminar course on "Security-Aware Processor Architecture Design" at the University of Virginia.