CacheBlitz: Reinforcement Learning for Cache Contention Attack Optimization

Shayan Chatiwala
Wayne Hills High School

Mentored by Mulong Luo
Florida State University

Introduction

- Microarchitectural side channels pose a serious threat to hardware security
  - Security evaluations require significant manual work
- **Cache contention attacks** exploit vulnerabilities in shared memory
  - Fuzzing methods struggle to identify efficient sequences
  - Past reinforcement learning (RL) tools limited to single-level caches and guessing secret victim addresses

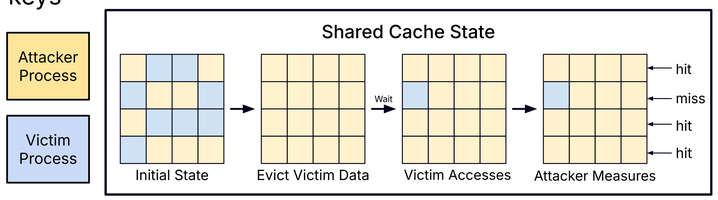
- **CacheBlitz** framework uses RL to discover short cache contention attack sequences for realistic memory hierarchies
  - Exploits cache replacement policy to evict victim data
  - Adaptable to **multi-level, multi-core** cache configurations
  - Generates successful sequences with up to **67% less instructions** than SOTA fuzzing

Background: Cache Contention Attacks

- Cache contention attacks exploit differences in access latencies between cache hits and misses to extract info about a victim process

- Typical attack formulation:
  1. Perform accesses to **evict** victim data from shared cache
  2. **Wait** for victim to perform access
  3. **Measurements** to detect location of victim data

- Many repetitions can be used to uncover secret encryption keys



Background: Reinforcement Learning

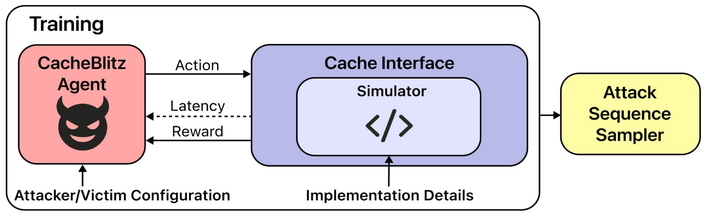
- **Agent** discovers action sequences that maximizes **reward**
- Receives **observations** and rewards from **environment**

System Design

- **Agent** - "plays" as attacker to generate attack sequences
  - **Actions**: attacker data access, trigger victim access
- **Environment** - simulated cache; support for two CPU cores, three cache levels, and LRU, Pseudo-LRU, BRRIP replacement policies
- **Observations** - hit/miss status returned after each access
- **Rewards -** large reward for successful evictions, small penalty for each action

- Uses PPO as RL algorithm; transformer as architecture

- Users can sample sequences after training has completed

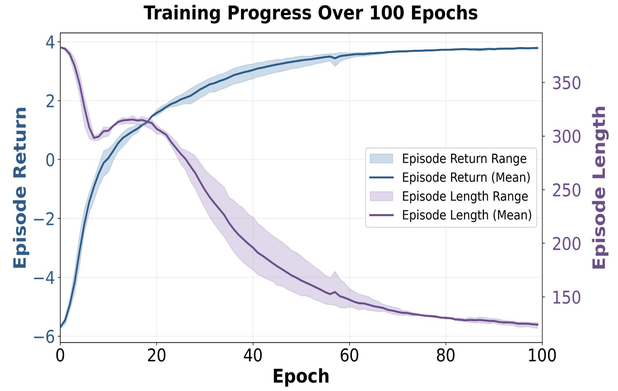


Results: CacheBlitz Training Progress

- CacheBlitz successfully evicts victim data 5 times per episode across four two-core, three-level cache configurations
- Figure shows training progress across three trials on BRRIP, 16-way LLC configuration

Table 1: Convergence Statistics for Four Configs

Cache Config	Epochs to Converge	Final Episode Length
LRU, 12-way LLC	21	73.69
LRU, 16-way LLC	60	89.93
BRRIP, 12-way LLC	29	80.46
BRRIP, 16-way LLC	91	110.24



Results: Comparison with Fuzzing

- PrimeTime, the mutation-based fuzzer used to discover Prime+Scope attacks was used for comparison (CCS '21)
- Both tools were used to construct sequences that prepare a cache for monitoring by the attacker - most nontrivial step of attack because it is oftentimes specific to replacement policy

Table 2: Attack Sequence Comparison: Instruction Counts

Cache Config	Method	# Instructions	PrimeTime Encoding
LRU, 12-way LLC	PrimeTime	41	R1_S4_P321S001230123
	CacheBlitz	16	
LRU, 16-way LLC	PrimeTime	54	R1_S4_P321S001230123
	CacheBlitz	18	
BRRIP, 12-way LLC	PrimeTime	41	R1_S4_P32S1032100123
	CacheBlitz	14	
BRRIP, 16-way LLC	PrimeTime	54	R1_S4_P012332100123S
	CacheBlitz	19	

- CacheBlitz demonstrates targeted exploitation of each LLC's replacement policy through minimal cache accesses
- By contrast, PrimeTime sequences contain redundancy
  - "Over-primes" the cache to guarantee eviction of victim's data
- Final sequences from both achieve 100% success rates in evicting victim data
- CacheBlitz sequences require 61.0% to 66.7% fewer instructions than PrimeTime

Table 3: Virtual CPU Cycle Count Comparison

Cache Config	PrimeTime	CacheBlitz
LRU, 12-way LLC	4,754	4,924
LRU, 16-way LLC	6,140	5,904
BRRIP, 12-way LLC	4,746	4,592
BRRIP, 16-way LLC	6,186	5,908

- To measure execution times, sequences were run on a cache simulator configured with typical Intel microprocessor access latencies for each memory level
- Only 2.3% fewer virtual CPU cycles on average
  - PrimeTime performs many **redundant memory accesses** → are hits that complete in just a few cycles

- Key advantages of CacheBlitz's shorter sequences
  - Reducing the attacker's memory footprint and instruction cache pressure
  - Achieving **higher instruction quality** where each operation contributes more directly to cache contention
  - CacheBlitz attacks are **more subtle and difficult to detect** through instruction-level monitoring or performance counter analysis