Advanced International Journal for Research (AIJFR)
E-ISSN: 3048-7641 ● Website: www.aijfr.com ● Email: editor@aijfr.com

Invisible Threats in the Cloud: Side-Channel and Shared-Resource Attacks

Abhijit Dadaso Dombale[1], Prof. Pooja Tupe[2]

[1,2] Department of Information Technology
University of Mumbai, Maharashtra, India

Abstract

Cloud computing environments rely heavily on multi-tenancy and shared hardware resources to provide scalable and cost-efficient services. However, shared-resource architectures introduce hardware-level security risks that traditional software isolation mechanisms cannot fully prevent. This paper presents a practical Proof-of-Concept (PoC) implementation of a cache-based Prime+Probe side-channel attack in a shared-resource environment using Windows Subsystem for Linux 2 (WSL2). The study demonstrates how an attacker process can detect victim activity by monitoring cache timing variations through shared CPU cache behavior. The attacker and victim processes were executed on the same physical processor to simulate cloud co-residency conditions. Experimental results showed clear timing spikes ranging from 160–220 CPU cycles during victim activity compared to stable idle-state timings of 80–120 cycles. The findings highlight the limitations of logical isolation in protecting against hardware-level information leakage and emphasize the growing importance of secure cloud architecture design, cache isolation, and hardware-aware defense mechanisms. The research also demonstrates how low-cost consumer hardware can reproduce realistic cloud side-channel behavior for cybersecurity experimentation and academic analysis.

Keywords: Cloud Security, Side-Channel Attack, Prime+Probe, Cache Timing Attack, Multi-Tenancy, Shared Resources, WSL2, Cybersecurity

1. Introduction

Cloud computing has transformed modern computing by enabling organizations to access scalable infrastructure and services on demand. Public cloud providers such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform allow users to deploy virtual machines, applications, and storage systems without maintaining physical hardware. The success of cloud computing largely depends on multi-tenancy, where multiple users share the same physical hardware resources.

Although virtualization technologies provide logical separation between tenants, hardware resources such as CPU caches, memory controllers, and shared buses remain physically shared. This creates opportunities for side-channel attacks, where attackers infer sensitive information indirectly by observing hardware behavior such as timing differences, cache access patterns, or resource contention.

Among various side-channel techniques, cache-based attacks such as Prime+Probe have become highly significant in cloud environments. These attacks exploit shared cache memory to monitor victim activity without directly accessing protected memory regions. Even strong encryption algorithms can become vulnerable when hardware-level leakage reveals execution patterns.

This research focuses on implementing a practical Proof-of-Concept (PoC) cache side-channel attack using the Prime+Probe technique. The experiment was performed in a shared-resource environment using Windows Subsystem for Linux 2 (WSL2) to simulate cloud co-residency conditions. The objective is to demonstrate how hardware-level timing analysis can bypass logical isolation mechanisms.

1.1 Problem Statement

Modern cloud systems rely on shared hardware infrastructure for efficiency and scalability. However, hardware sharing creates security risks because CPU caches and memory systems are not fully isolated between tenants. Traditional software-based isolation methods cannot completely prevent side-channel leakage. Attackers may exploit timing variations and cache behavior to monitor victim activity or infer sensitive information.

1.2 Research Objectives

The main objectives of this research are:

1. To study the security risks associated with multi-tenant cloud environments.
2. To implement a cache-based Prime+Probe side-channel attack.
3. To analyze timing variations caused by cache evictions.
4. To evaluate the effectiveness of cache timing analysis in shared-resource systems.
5. To discuss possible mitigation and defense mechanisms.

1.3 Scope of the Study

This study focuses on cache-based side-channel attacks in shared-resource environments. The implementation is limited to a local Proof-of-Concept setup using WSL2 and Ubuntu Linux. The research primarily analyzes cache timing behavior and does not target real-world cloud providers or attempt unauthorized access to external systems.

1.4 Limitations

The experiment was conducted on a single physical laptop instead of a real cloud infrastructure. Background system processes may introduce measurement noise. The study focuses only on Prime+Probe attacks and does not cover all categories of side-channel attacks.

2. Literature Review

Cloud computing security has become a major area of research due to the widespread adoption of virtualization and shared-resource architectures. Researchers have shown that hardware-level resource sharing introduces vulnerabilities that traditional software isolation cannot completely prevent.

Ristenpart et al. demonstrated that attackers could achieve co-residency in cloud environments and exploit shared resources to extract sensitive information. Yarom and Falkner introduced the Flush+Reload attack, which showed how shared cache memory can leak cryptographic operations through timing analysis. Osvik, Shamir, and Tromer further demonstrated cache attacks against AES encryption systems.

Research on modern virtualization technologies such as Xen, KVM, Docker, and Kubernetes indicates that logical isolation is often insufficient when hardware resources remain shared. Studies have also explored memory deduplication attacks, timing attacks, and speculative execution vulnerabilities such as Meltdown.

Recent literature emphasizes the growing challenge of detecting side-channel attacks because these attacks often appear as normal system behavior. Researchers have proposed mitigation strategies including cache partitioning, hardware enclaves, timer restrictions, and machine-learning-based detection systems.

The literature confirms that side-channel attacks remain a critical security concern in modern cloud environments, especially where multi-tenancy and high-performance resource sharing are common.

Although significant research has been conducted on cache side-channel attacks in enterprise cloud infrastructures, limited practical educational implementations exist using lightweight environments such as WSL2. This research attempts to bridge that gap by providing a simplified yet realistic Proof-of-Concept implementation for academic and experimental purposes.

3. Methodology

3.1 Experimental Design

The research adopted a practical experimental approach to demonstrate a cache-based side-channel attack. The experiment was conducted using Windows 11 with Windows Subsystem for Linux 2 (WSL2) running Ubuntu Linux. This environment allowed both attacker and victim processes to execute on the same physical processor and share the same cache hierarchy.

3.2 System Configuration

The implementation environment consisted of:

Component	Specification
Operating System	Windows 11
Virtualization Platform	WSL2 Ubuntu
Processor	Intel Core i3-8130U
RAM	8 GB
Programming Languages	C and Python
Visualization Tool	Matplotlib
Attack Framework	CacheSC

3.3 Prime+Probe Technique

The Prime+Probe attack works in three stages:

1. Prime Phase – The attacker fills selected cache sets with its own data.
2. Victim Execution – The victim process performs memory operations that may evict attacker cache lines.
3. Probe Phase – The attacker measures cache access times to determine whether eviction occurred.

Timing measurements were collected using the RDTSC instruction, which provides high-resolution CPU timing information.

3.4 Victim Process

A Python-based victim script was developed to generate cache activity. The victim repeatedly created large arrays and performed summation operations to trigger cache evictions and resource contention.

3.5 Data Collection

Timing data generated by the attacker process was stored in log files. The collected data was processed using Python and Matplotlib to generate timing graphs for analysis.

4. Experimental Setup and Results

4.1 Implementation Details

The attack implementation used the CacheSC library with custom configuration changes based on the processor cache specifications. The src/device_conf.h configuration file was modified to optimize cache associativity parameters for the Intel i3-8130U processor.

The attacker process continuously monitored cache access timing while the victim process generated memory-intensive operations.

4.2 Execution Procedure

The experiment followed these steps:

1. Configure and compile the CacheSC attack framework.
2. Launch the attacker process.
3. Start the victim process after establishing baseline timing measurements.
4. Monitor timing variations caused by cache evictions.
5. Generate timing graphs using Matplotlib.

4.3 Results

The experiment produced consistent and observable timing spikes during victim activity.

Table 1: Cache Timing Comparison

Run Number	Idle Cache Cycles	Victim Active Cycles
Run 1	92	185
Run 2	88	201
Run 3	110	220
Run 4	95	198

The timing spikes indicate successful cache eviction caused by the victim process. When the victim became active, cache access latency increased significantly, confirming that shared cache behavior can leak information about neighboring processes. During baseline execution, cache access timings remained relatively stable with minimal fluctuations. However, when the victim process initiated memory-intensive operations, measurable latency spikes were observed due to eviction of attacker-controlled cache lines. These timing variations confirm that shared cache activity can indirectly reveal victim behavior through the Prime+Probe technique

4.4 Observations

Several important observations were made:

- Cache timing spikes were clearly visible during victim execution.
- Shared hardware resources allowed indirect monitoring of victim activity.
- Logical isolation provided by the operating system could not prevent hardware-level leakage.
- Noise from background processes slightly affected timing consistency.

4.5 Challenges Faced

The implementation process involved multiple technical challenges:

- Dependency and compilation errors during CacheSC installation.
- WSL2 configuration issues.
- Cache parameter tuning for the Intel processor.
- Noise reduction for stable timing measurements.
- Graph generation and Python environment setup.

In addition to demonstrating the attack, this research implemented a lightweight visualization-based monitoring approach using Python and Matplotlib to observe cache timing fluctuations in real time. This simplified visualization method improves educational understanding of side-channel behavior and provides an accessible platform for future cybersecurity experimentation.

5. Discussion

The results confirm that Prime+Probe side-channel attacks can successfully monitor victim activity through shared cache behavior. Even though the experiment was performed locally, the setup closely resembles co-residency scenarios found in cloud computing environments.The observed timing spikes demonstrate that hardware-level information leakage remains possible despite software isolation mechanisms. This highlights the limitations of current virtualization security models.

Modern cloud infrastructures prioritize performance and resource sharing, which increases the risk of cache-based attacks. Techniques such as cache partitioning, hardware isolation, and confidential computing may help reduce these risks, but they often introduce performance overhead.

The findings are particularly relevant for cloud platforms hosting sensitive applications such as banking systems, cryptographic services, healthcare infrastructure, and confidential enterprise workloads. Even without direct memory access, attackers may still infer operational behavior through shared hardware resources, emphasizing the importance of hardware-aware cloud security mechanisms.

6. Conclusion

This research successfully demonstrated a cache-based Prime+Probe side-channel attack in a shared-resource environment using WSL2. Experimental results showed clear timing variations caused by cache eviction during victim activity.

The study highlights that logical isolation alone is insufficient to protect against hardware-level side-channel leakage. As cloud computing continues to grow, stronger hardware-aware security mechanisms will become essential for protecting sensitive workloads. The research also demonstrates that consumer-grade hardware can be used to reproduce realistic side-channel attack behavior, emphasizing the practical relevance of these threats.

7. Future Scope

Future work can extend this research by:

- Implementing attacks in real multi-VM cloud environments.
- Studying hypervisor-level cache isolation mechanisms.
- Developing machine-learning-based side-channel detection systems.
- Integrating real-time visualization dashboards using Streamlit.
- Evaluating modern defenses such as Intel CAT and confidential computing.

References

1. Y. Yarom and K. Falkner, “Flush+Reload: A High Resolution, Low Noise, L3 Cache Side-Channel Attack,” *Proceedings of the USENIX Security Symposium*, 2014.
2. D. A. Osvik, A. Shamir, and E. Tromer, “Cache Attacks and Countermeasures: The Case of AES,” *Proceedings of the CT-RSA Conference*, 2006.

3. Y. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Cross-VM Side Channels and Their Use to Extract Private Keys," *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2012.
4. T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, You, Get Off of My Cloud," *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, 2009.
5. M. Lipp et al., "Meltdown: Reading Kernel Memory from User Space," *Proceedings of the USENIX Security Symposium*, 2018.
6. A. Barenghi et al., "A Survey of Side-Channel Attacks and Their Countermeasures," *IEEE Communications Surveys & Tutorials*, 2012.
7. Z. Wang and R. B. Lee, "New Cache Designs for Thwarting Software Cache-Based Side Channel Attacks," *Proceedings of the International Symposium on Computer Architecture (ISCA)*, 2007.
8. D. Gruss et al., "Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript," *Proceedings of the Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, 2016.
9. T. Kim et al., "StealthMem: System-Level Protection Against Cache-Based Side Channel Attacks in the Cloud," *Proceedings of the USENIX Security Symposium*, 2012.
10. S. Crane et al., "Thwarting Cache Side-Channel Attacks Through Dynamic Software Diversity," *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2015.