A Noise-resilient Detection Method against Advanced Cache Timing Channel Attack

Hongyu Fang, Sai Santosh Dayapule, Fan Yao, Miloš Doroslovački, Guru Venkataramani

Department of Electrical and Computer Engineering
The George Washington University
Washington, DC, USA
{hongyufang_ee, saisantoshd, albertyao, doroslov, guruv}@gwu.edu

Abstract—Recent researches show that computers which are physically shared by multiple users are vulnerable to microarchitecture-based information leakage. Among all microarchitecture components, cache provides the largest attack surface. Cache timing channels manipulate the cache access latency to leak information leaving no physical trace. To mitigate cache timing channels, various detection methods are proposed. However, with the knowledge of existing detection methods, an advanced adversary can intentionally inject noise to evade detection. For example, the detection based on correlation method which extracts the repetitive behavior of cache timing channels can be evaded by randomizing the gap between information transmitting and receiving activity. The classification based detection would be obfuscated if adversary imitate the behavior of benign applications. We propose a novel noise-resilient detection method which focuses on the dependency between behavior of two processes. For each process, we define a group of events and track the conditional probability of every event given the appearance of the events from another process. With this method, we are able to detect the existence of cache timing channels. Our detection method is hard to evade because the dependency of cache behavior is necessary for any communication through cache timing channels.

I. CACHE TIMING CHANNEL

With the development of cloud computing, more and more developers deploy their processes and data on shared platforms to reduce infrastructure cost and boost performance. The scenario where multiple processes from different owners running on a same physical machine becomes ubiquitous. To prevent information leakage, platform operators usually prohibit direct communication of processes from different user domains using Operating Systems or Hypervisors. However, as long as the hardware is still shared, the microarchitecture covert and side channels are still potential security hazard for information leakage. Among all microarchitecture covert and side channels, cache timing channel [8], [11], [12], [13], [15], [16] is one of the most notorious because cache provides the largest attack surface and the channels cannot be mitigated by software-based approach. Cache timing channel usually involves two processes: trojan/victim and spy. For the side channel scenario, a victim unconsciously leaks information through its cache access pattern while a malicious spy tries to track victim’s cache access pattern and finally reveals the victim’s secret. In a covert channel scenario, a trojan which has access to secret tries to leak information intentionally by modulating cache access and the spy manages to receive trojan’s information by observing its cache access pattern. We note that the only difference between trojan and victim is whether the process leaks information on purpose. The behavior of trojan and victim is simillar. In the following paper, we use the word trojan to represent both trojan in the covert channel scenario and victim in the side channel scenario.

Among various implementation of cache timing channels, the prime+probe is the most common protocol because it does not have prerequisites beyond physically shared cache. In prime+probe attack, the trojan manipulate the spy’s cache access latency by creating conflict miss. Conflict miss is caused by cache block replacement. When one process accesses a memory line from DRAM which is mapped to a full cache set, the newly accessed memory line would be brought to cache and replace one cache block inside the cache set. If a process access the replaced cache block, it would suffer from cache conflict miss and observe higher latency.

As shown in Figure 1 to launch prime+probe attack, the spy firstly primes every cache block in cache sets with its own memory lines, then the trojan transmits bit ‘1’ by evicting all cache blocks owned by spy and transmits bit ‘0’ by staying idle. After trojan’s activity, the spy probes the memory lines which were used to prime and measures the access latency. If the trojan transmits bit ‘1’, the spy would observe high latency because of conflict misses. If the trojan transmits bit ‘0’, the spy would observe low latency because all its memory lines remained in cache. From the observed cache access latency, the spy can infer the trojan’s behavior and decipher the information from trojan. The measured cache access latencies of spy during transmission are shown in Figure 2 the latencies
The existing detection methods mainly include two categories: 1. Correlation-based detection [14], [2] which aims at extracting the repetitive pattern of trojan and spy caused by information transmission, 2. Classification-based detection [7].

A. Existing Detection Methods and Limitation

To mitigate cache timing channels, various detection methods are proposed [14], [2], [7], [3], [4], [9], [10], [5], [6]. The existing detection methods mainly include two categories: 1. Correlation-based detection [14], [2] which aims at extracting the repetitive pattern of trojan and spy caused by information transmission, 2. Classification-based detection [7].

Fig. 3: Cache access pattern of different cache timing channel protocols

2. Classification-based detection [7] which focuses on comparing characteristics of suspicious process and baseline benign processes. These detection methods can efficiently detect adversaries which are not aware of the existence of detection. However, with the knowledge of detection strategy, the smarter adversaries could manage to evade detection by noise injection and timing randomization. The repetitive pattern would be obfuscated by timing randomization so that correlation-based methods would fail. With noise injection, the adversary can pretend a benign cache-intensive process to evade classification-based detection. To illustrate how can an advanced adversary evade detection, we implemented an advanced cache timing channel on Gem5 and compared the observed cache miss of advanced cache timing channel with the naive cache timing channel. The advanced spy and trojan exploit 16 cache sets to communicate using prime+probe protocol. To evade potential detection, the spy inflates its cache miss in another 16 cache sets. We note the cache sets exploited to communicate as "communication sets" and the cache sets for noise injection as "noise sets." Besides noise injection, spy also randomize the timing of cache access. The number of misses of trojan and spy are shown in Fig. 4a. As shown in Fig. 4a, the miss rates of naive trojan and spy are correlated. When trojan suffers from a number of cache misses, the spy also suffers from cache misses. When trojan doesn’t have cache miss, the spy would not suffer from cache miss either. The misses of advanced trojan and spy are shown in Fig. 4b. The spy’s misses are not correlated with trojan’s and its behavior is not repetitive compared to Fig. 4a. hence the accuracy of correlation-based detection would degrade. Besides, the spy can create additional noise to pretend to be a cache-intensive benign application so that the classification-based detection would fail to detect it.
set 2

\[ P(s_i | t_j) = \frac{\# \text{time windows with both labels: } S_i, T_j}{\# \text{time windows with label } T_j} \]

\[ P(s_i | T_j) = \frac{\# \text{time windows with both labels: } S_i, T_j}{\# \text{time windows with label } T_j} \]  

As shown in Figure 5b when the time interval is small enough such that at most one cache miss could happen within one time interval, the first conditional probability can be represented using cache miss rate:

\[ P(s_i | t_j) = R_{ij} t_{\text{window}} \]  

where \( R_{ij} \) is the number of spy’s cache misses in cache set \( i \) in time windows with label \( T_j \) divided by the time elapse of all time windows with label \( T_j \) and \( t_{\text{window}} \) is the size of each time window. The second conditional probability in (2) can be expressed in the similar way where \( R_{ij} \) is replaced by \( R_{ij} \).

We note that it is more practical to track cache miss rate in real machine. In the following paper, we will use cache miss rate to estimate conditional probabilities we mentioned in (1).

To illustrate the efficiency of detection, we do the cache miss rate analysis on the advanced cache timing channel introduced in Section II-A. Figure 6a shows the cache miss rate of spy given that trojan accessed the cache set 0 within 150-microsecond time interval. And the Figure 6b shows the cache miss rate of spy given that trojan did not access the cache set 0 within the 150-microsecond window. We can observe that the conditional probability on set 0, 5, 7, 8, 9, 10 and 11 change significantly when trojan has different behavior indicating they are communication sets. On the other hand, the conditional probabilities of the other sets do not change a lot when trojan access or not cache set 0, indicating that they are noise sets.
The cache miss rate of spy given that trojan accessed the cache set 0 within the 150-microsecond window.

The cache miss rate of spy cache miss given that trojan did not access the cache set 0 within the 150-microsecond window.

Fig. 6: Cache Miss Rate of spy in round-robin attack

Fig. 7: Histogram of cache miss rate decrease of spy when trojan does not access the cache set 0 within given time window.

III. EXPERIMENTAL SETUP

We implement the cache timing channel attack using Gem5 \cite{1}, a cycle-accurate, full-system simulator. We configure Gem5 with four x86 cores, 32 KB private L1 and 512 KB, 8-way shared L2 caches. All the experiments are run on full system mode under Linux kernel version 2.6.32. We collect the cache access traces from simulator and implement the detection methods on the traces.

We deploy the trojan and spy processes on separated CPU cores. The trojan and spy communicate in either round-robin or parallel fashion as we discussed in Section I. For the round-robin attack, the trojan and spy use 7 communication sets. The spy injects noise in 7 noise sets by creating self evictions. For the parallel attack, the trojan and spy use 14 communication sets to transmit information.

IV. EVALUATION

A. Round-robin Noisy Attack

As we discussed in Section II-B the cache miss rates of spy in communication sets drop significantly when trojan does not access communication sets while the cache miss rates in noise sets drops much less than those in communication sets. Figure 9 shows the histogram of cache miss rate decrease of spy given different behaviors of trojan. The miss rate decrease is normalized with the cache miss rate when the trojan accesses the communication sets within a given time interval. Cache miss rate drops 100% in communication sets which indicates the spy’s behavior is strongly dependent on trojan’s activity. The cache miss decrease in noise set ranges from -10% to 50% which is much lower than the cache miss rate decrease in communication sets. The miss rate decrease can efficiently distinguish communication sets from noise sets and finally capture the covert communication in noisy background. We note that as demonstrated in previous work \cite{2}, the benign applications would not have this correlated behavior, so our design is also able to identify adversary process when it is running with benign workloads.

B. Parallel Attack

The cache miss rates of spy given different trojan activity implementing parallel protocol are shown in Figure 8. The cache miss rate of spy in cache set 0 drops 60% when trojan does not access the same cache set in the last 150 microsecond. The cache miss rate decrease in other cache sets may drop less than the cache miss rate in cache set 0 because the activities of trojan and spy are not synchronized. When spy is scanning the cache set 1 to 13, it is possible that the trojan has already start next transmission in cache set 0. For parallel attack, examining the spy’s cache miss rate given different trojan’s activity in the same cache sets can still distinguish the adversary behavior.
V. CONCLUSION

In this paper, we propose a novel noise-resilient detection method based on conditional probability to prevent information leakage through cache timing channel. We demonstrated our work on both round-robin and parallel cache timing channel protocol. Our experiments show that the proposed method is able to distinguish cache sets which are exploited by adversary in noisy background by calculating cache miss rate in different condition.

VI. ACKNOWLEDGMENT

This manuscript is based on work supported by the US National Science Foundation under CAREER Award CCF-1149557 and grant CNS-1618786, and Semiconductor Research Corp. (SRC) contract 2016-TS-2684. Any opinions, findings, conclusions, or recommendations expressed in this article are those of the authors, and do not necessarily reflect those of the NSF or SRC.

REFERENCES


