Survey of CPU Cache-Based Side-Channel Attacks: Systematic Analysis, Security Models, and Countermeasures

Authors: Chao Su and Qingkai Zeng
Sources: Security and Communication Networks, 2021
Speaker: Chin-Yu Sun
Outline

• **Introduction**
  • Analysis of the attacks
  • Analysis of the defenses
  • Challenges and trends
  • Conclusion
Cache

• A modern CPU cache is designed with a hierarchy architecture:
  • L1, L2, and L3 (Last-Level Cache, LLC)

• Cache capacity and access latency

---

Cache-based side-channel attack

• By **observing** the changes in CPU cache states
  • Attackers infer the memory usage of the victim program and thus leak sensitive information

• Hard to detect
  • Traditional intrusion detections cannot respond effectively

Victim  ![Access (or not)](image)

Attacker  ![Observing](image)

1. Access
2. No access
Trend (1/2)

- Research (Web of Science)

Figure 1: Number of research papers on side-channel attacks from 2012 to 2019 from Web of Science.
Trend (2/2)

• Attack’s ability
  • The attacker and the victim need to stay in the same process [2]
  • Sensitive information can be stolen across VMs
  • Attackers can even bypass Intel SGX and steal the secret from enclaves [3]

• OpenSSL
  • RSA modules
  • AES
  • DSA (Digital Signature Algorithm)
  • ECC
  • Others
Contributions

• Concludes the general workflow of the attacks
• Establishes an attack model with four factors:
  • Vulnerability
  • Cache type
  • Pattern
  • Range
• Conducts a comprehensive analysis of different defense strategies
Outline

• Introduction
• Analysis of the attacks
• Analysis of the defenses
• Challenges and trends
• Conclusion
1. Define the **connection** between the victim and the attacker
   - Search for the correlation in the cache
     - Shared library (OpenSSL 0.9.8n) [19]
     - Large page mechanism (VMware and Xen) [20]
2. Collect the activities in the cache of the attacker’s program while it’s running
   • **Pre-set** the state of the cache
     • Full: continuous memory reading or writing
     • Empty: CLFLUSH instruction
3. **Speculate** on the cache changes of the victim program
   - The cache behaviors of the victim program can be inferred from the cache states
     - Consistency: share the same cache states (hit or miss) [5, 6, 16, 20–23]
     - Exclusion: exclusive use of the cache mutually
4. **Infer** the sensitive information of the victim’s program
   - To define the association between state changes and sensitive information of the victim
     - Square-and-multiply algorithm (OpenSSL 0.9.7c - RSA module)
Threats of the attacks

1. Disclosure of sensitive information such as privacy
2. Deliver the results of malicious code execution [17]
3. Denial of service [24-26]
   • Cause a performance reduction by 95% and overhead increase by 7.9X
Model of the attacks

- Four key elements that constitute the cache side-channel attack model
  - Program vulnerability
  - Cache type
  - Pattern
  - Range
Program vulnerability

• Refers to the vulnerable code inside the victim program that can be compromised
  • CVE-2018-0737 [27]
    • The OpenSSL RSA Key generation algorithm has been shown to be vulnerable to a cache timing side channel attack. An attacker with sufficient access to mount cache timing attacks during the RSA key generation process could recover the private key. Fixed in OpenSSL 1.1.0i-dev (Affected 1.1.0-1.1.0h). Fixed in OpenSSL 1.0.2p-dev (Affected 1.0.2b-1.0.2o).
  • CVE-2017-5754 [28]
  • CVE-2016-2178 [29]
Cache type

• L1-I [6, 21]
  • The instruction cache, which is specially used to store instructions in the first-level cache
  • Whether instructions in cache or not indicates whether it has been recently accessed

• L1-D [2, 5, 14, 23]
  • Stores data
  • The data structure of the victim’s program and the access pattern of variables can also be utilized to infer sensitive data

• LLC [4, 8, 9, 12, 13, 17, 20, 22, 30-36, 38-41]
  • Shared
  • Larger access latency (better robustness when using LLC as their carrier)
Pattern

1. Prime + Probe
2. Flush + Reload
3. Evict + Reload
4. Evict + Time
5. Flush + Flush
6. Invalidate + Transfer [16]
   - Across processors
Different types of cache have their own scope of influence, authors summarize the current attacks into four levels:

1. **Core**
   - L1 cache: only programs executed on the same physical core can access the same L1 cache at the same time

2. **Package**
   - The package range is shared between objects within the same package

3. **NUMA**
   - Same memory controller, smaller than the system sharing

4. **System**
   - The resources that all processes in the system can access
## Research studies

<table>
<thead>
<tr>
<th>Attacks</th>
<th>Patterns</th>
<th>Cache types</th>
<th>Range</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>[14]</td>
<td>Prime + Probe</td>
<td>L1-D</td>
<td>Core</td>
<td>OpenSSL (0.9.7c) RSA</td>
</tr>
<tr>
<td>[2]</td>
<td>Prime + Probe</td>
<td>L1-D</td>
<td>Core</td>
<td>AES</td>
</tr>
<tr>
<td>[5, 23]</td>
<td>Prime + Probe</td>
<td>L1-D</td>
<td>Core</td>
<td>OpenSSL (0.9.8) AES</td>
</tr>
<tr>
<td>[21]</td>
<td>Prime + Probe</td>
<td>L1-I</td>
<td>Core</td>
<td>OpenSSL (0.9.8d) RSA</td>
</tr>
<tr>
<td>[6]</td>
<td>Prime + Probe</td>
<td>L1-I</td>
<td>Core</td>
<td>OpenSSL (0.9.8e) RSA</td>
</tr>
<tr>
<td>[8]</td>
<td>Evict + time</td>
<td>LLC</td>
<td>CPU</td>
<td>Kernel memory</td>
</tr>
<tr>
<td>[20]</td>
<td>Flush + Reload</td>
<td>LLC</td>
<td>CPU</td>
<td>OpenSSL (0.9.8n) AES</td>
</tr>
<tr>
<td>[30]</td>
<td>Flush + Reload</td>
<td>LLC</td>
<td>CPU</td>
<td>GnuPG (1.4.13) RSA</td>
</tr>
<tr>
<td>[4]</td>
<td>Flush + Reload</td>
<td>LLC</td>
<td>CPU/VM</td>
<td>OpenSSL (1.0.1f) AES</td>
</tr>
<tr>
<td>[31–34]</td>
<td>Flush + Reload</td>
<td>LLC</td>
<td>CPU</td>
<td>OpenSSL (1.0.1e) ECDSA</td>
</tr>
<tr>
<td>[9]</td>
<td>Flush + Reload</td>
<td>LLC</td>
<td>CPU/VM</td>
<td>User activities</td>
</tr>
<tr>
<td>[35]</td>
<td>Flush + Reload</td>
<td>LLC</td>
<td>CPU</td>
<td>TLS and DTLS</td>
</tr>
<tr>
<td>[36]</td>
<td>Flush + Reload</td>
<td>LLC</td>
<td>CPU</td>
<td>Keyboard input</td>
</tr>
<tr>
<td>[37]</td>
<td>Prime + Probe</td>
<td>LLC</td>
<td>CPU</td>
<td>GnuPG (1.4.13/1.4.18) ElGamal</td>
</tr>
<tr>
<td>[22]</td>
<td>Flush + Reload</td>
<td>LLC</td>
<td>CPU</td>
<td>OpenSSL DSA</td>
</tr>
<tr>
<td>[38]</td>
<td>Prime + Probe</td>
<td>LLC</td>
<td>CPU</td>
<td>AES</td>
</tr>
<tr>
<td>[39]</td>
<td>Flush + Flush</td>
<td>LLC</td>
<td>Core</td>
<td>AES T-table</td>
</tr>
<tr>
<td>[16]</td>
<td>Invalidate + Transfer</td>
<td>—</td>
<td>System</td>
<td>Open SSL AES/libgcrypt ElGamal</td>
</tr>
<tr>
<td>[12, 13, 17]</td>
<td>Flush + reload/evict + time</td>
<td>LLC</td>
<td>CPU</td>
<td>Memory isolation</td>
</tr>
<tr>
<td>[40, 41]</td>
<td>Prime + Probe</td>
<td>LLC</td>
<td>CPU/enclave</td>
<td>RSA</td>
</tr>
</tbody>
</table>
Outline

• Introduction
• Analysis of the attacks
• Analysis of the defenses
• Challenges and trends
• Conclusion
Defenses strategies

• Information Independency
• Time Blinding
• Time Sharing
• Resource Isolation
• Anti-Co-Resident Detection
• Channel Interference
Information independency

• Attack requires a correlation between the sensitive information and the cache states
  • **Constant time** ensures that the behavior of the target program is completely independent of sensitive data
    • Turn the execution time of different branches in OpenSSL into **constants** that do not depend on inputs [47, 48]
  • CVE-2018-0737 [27], CVE-2018-0734 [43], CVE-2018-12438 [44], CVE-2014-0076 [45], and CVE-2016-2178 [29] modify codes in different branches, making the program **execution sequence unrelated to the key**
  • Not to perform **sensitive information-dependent access operations** on the memory that exceeds the granularity of the cache line [46]
Time blinding

• Modifying the time (CPU cycles) read by the attacker can affect the judgment of the cache state

  1. Virtual time
     • hides the real access time
  2. Time black box
     • treats the entire system as a whole
Time sharing

• The victim and attacker programs use the cache in turn [52]
  • Clears all cache contents when different VMs are switched
  • Brings huge performance overhead to the system
    • Clearing the contents of the L1 cache will cause a 17% performance cost, If the LLC is cleared out of the cache every time the VM is switched, the performance overhead will be even greater [53]
Resource isolation

• Turning off the Simultaneous multithreading (SMT) [54]
• Turning off the page sharing [53]

• Cache partitions
  • Hardware isolation [55, 56]
  • Software isolation [57]
Anti-co-resident detection

• The attacker can only carry out a complete side-channel attack when it is co-resident in the same environment with the victim
  • Co-resident detection methods [15, 58-61]

• Prevented from inferring co-resident information
  • Optimizing virtual machine isolation and hardware resource management
  • Modifying the network configuration
Channel interference

• Injecting **noise** into the cache can interfere with channel usage
  • Fuzzy time [49, 62]
  • Compress, reload, and randomize the lookup table [63]
  • Random permutation cache (RPcache) [56]
  • Set up a bystander virtual machine to inject noise into the L2 cache [64]
    • By Continuous Time Markov Process
Outline

• Introduction
• Analysis of the attacks
• Analysis of the defenses
• Challenges and trends
• Conclusion
Challenges

• Side-channel attacks use CPU cache as a carrier, which is hardly paid attention in developing
• Unwilling to completely isolate hardware resources
• Artificial Intelligence helps to recognize the pattern of cache activities
Trends

• Developers should avoid connecting sensitive information with cache states during the execution of the programs
• Deploy more efficient resource isolation
• The diversification of obfuscation and identification of the cache
Outline

• Introduction
• Analysis of the attacks
• Analysis of the defenses
• Challenges and trends
• Conclusion
Conclusion

• Survey paper
• Systematic analysis
• Summarize
  • Attack workloads
  • Security models
  • Countermeasures