# Research on Microarchitectural Cache Attacks

Yao Lu [a], Kaiyan Chen [b], Yinlong Wang [c]

Simulation Center of Ordnance Engineering College Army Engineering University Shijiazhuang, Hebei Province, China

[a]294102746@qq.com, [b]chen_wu2013@163.com, [c]returnsjz@sina.com

**Abstract.** This paper summarizes the basic concepts and development process of cache side-channel attack, analyses three basic methods (Evict and Time, Prime and Probe, Flush and Reload) from four aspects: Attack conditions, realization process, applicability, and characteristics, then I expound how to apply side-channel attack methods on CPU vulnerability.

**Keywords:** Side-channel attacks; Cache; CPU vulnerability; Microarchitecture.

## 1. Background

Cryptography is a technique used to confuse plaintext [1], It transforms normally identifiable information (plaintext) into unrecognizable information (ciphertext). At the same time, the encrypted ciphertext can be transferred back to the normal information through the key, the privacy information of users at this stage is mostly realized by encryption technology [28], so the security of personal information depends on the security of the encryption algorithm.

### 1.1 Cryptographic Algorithms

Cryptographic algorithms have always been an important research object in cryptography, in recent years has also been rapid development, these algorithms are: RIJINDAEL, MARS, RC6, Twofish, Serpent, IDEA, CS-Cipher, MMB, CA-1.1, SKIPJACK Symmetric cryptographic algorithms such as Karn and backpack public key cryptography, RSA, ElGamal [29], ECC [19], NTRU and other asymmetric cryptographic algorithms [27]. in the opinion of the development trend of international mainstream cryptographic algorithms at present [25]:

The symmetric cryptographic algorithm transitions from DES-3 to AES, and the password length is gradually increased: 128, 192, 256.

An asymmetric cryptographic algorithm RSA [2] will be used for a long period of time, gradually increasing the length from the length 2048, and ECC [19] further algorithm development and gradually dominant position, the secret key length is increased gradually from 224 bit.

Hash algorithm MD5 is gradually reduced for security reasons, SHA-1 will gradually transition to SHA-2, SHA-3.

Lightweight passwords have been a research focus, symmetric block ciphers (such as DESL, SEA, HIGHT), symmetric cipher stream ciphers (such as Rabbit, Hc-128, hardware-based MICKEY 2.0), asymmetric ciphers (such as BlueJay, NTRU), Hash algorithm (such as H-PRESENT- 128, C-PRESENT-192, Gluon, QUARK).

Table 1 lists the application scenarios and security comparisons of some single-key encryption algorithms. some combination of integrated application of technology is the use of a variety of algorithms, such as EMV2000, a combination of asymmetric and symmetric algorithm, offline authentication using asymmetric algorithms, take symmetric algorithms while online transactions.

Table 1. encryption algorithm application scenarios, security comparison

| name | Method to realize | calculating speed | safety | improvement measures | Application |
|---|---|---|---|---|---|
| DES | 40-56bit key | general | Fully dependent on the key, vulnerable to exhaustive search | Double, triple DES, AES | Suitable for hardware implementation |
| IDEA | 128-bit key 8 iterations | Slower | Military grade, resistance analysis and correlation analysis | The length of the word is 32 bits and the key is 256 bits. It uses 232 mode plus 232+1 mode multiplication. | Suitable for ASIC design |
| GOST | 256bit key 32 iterations | Faster | Military level | Increase the number of iterations | S box can be randomly selected to facilitate software implementation |
| Blowfish | 256-448bit Key, 16 rounds of iteration | Fastest | Military-grade, you can adjust security by changing the key length | | Suitable for fixed key occasions, not suitable for frequently changing keys and smart cards |
| RC4 | Variable key length | Fast DES ↓ L0 times | Immune to differential and linear attacks, highly nonlinear | Key length is relaxed to 64bit | Simple algorithm, easy to program |
| RC5 | Key length and number of iterations are variable | Speed can be selected based on the values of the three parameters | Anti-linear attack with more than six rounds, compromise between security and speed by adjusting word length, key length and iteration number of rounds | Introducing data | Suitable for microprocessors of different word lengths |
| CAST128 | Variable key length, 16 rounds of iteration | Faster | Resistant to linear and differential attacks | Increase the key length to form CAST256 | Suitable for PC and UNIX workstations |

- DES (Data Encryption Standard): Symmetric algorithms/fast/suitable for large collections of data encryption.
- 3DES (Triple DES): Based on DES, encrypting a block of data three times with three different keys which was more powerful.
- RC2 and RC4: A symmetric algorithm which encrypted large amounts of data with a variable length key, faster than DES.
- IDEA (International Data Encryption Algorithm) International Data Encryption Algorithm, using 128-bit keys to provide very strong security.
- RSA: Invented by RSA Security, which support variable public key length, the length of the file date to be encrypted is also variable, an asymmetric algorithm.
- DSA (Digital Signature Algorithm): Digital Signature Algorithm is a standard DSS (Digital Signature Standard), strictly speaking, not an encryption algorithm.
- AES (Advanced Encryption Standard): Advanced Encryption Standard, a symmetric algorithm, is the next generation of encryption algorithm standard, fast, high security level, Rijndael algorithm which take the AES standard is the main algorithm of 21st century.
- BLOWFISH: it uses variable-length key which could up to 448, it runs very quickly.
- MD5: strictly it is not an encryption algorithm, just a summary algorithm.
- PKCS: The Public-Key Cryptography Standards (PKCS) is a set of public key cryptography standards developed by RSA Data Security and its partners in the United States, including certificate applications, certificate updates, certificate invalidation tables, and extended certificate content. And a series of related agreements on digital signatures, format of digital envelopes.
- SSF33, SSF28, SCB2 (SM1): The national crypto bureau's covert commercial algorithms.
- ECC (Elliptic Curves Cryptography): Elliptic Curve Cryptography.
- TEA (Tiny Encryption Algorithm): encryption algorithm is simple and efficient, fast encryption and decryption, simple. But it’s security can’t compare DES.

### 1.2 Channel Attacks

In 1996, Kocher [3] officially proposed the idea of Side-channel Attacks. Kocher believes that electronic devices will leak sensitive information through various channels (also called channel, such as sound waves, electromagnetic radiation, power consumption, timing.) when performing cryptographic operations. The channel attack method is to collect and analyze these information lines. And fetch information related to cryptographic operations.

In 1998, Kesley [4] and others first proposed the idea that Cache hit rate can be used for key analysis, which subvert the traditional key analysis method (the legacy trace in storage devices is also the source of information leakage), which has aroused widespread concern of researchers. After that, the researchers used data cache and instruction cache as research target, several feasible channel attacks methods are proposed. which take a serious threat to the security of the algorithm. But most of the research results are based on a stand-alone non-virtualized environment.

Until 2009, Ristenpart et al. first proposed a security threat across the virtual machine Cache channel attack in the cloud environment, and used the Prime-Probe method detect the Cache load status information and user's keystroke interval information of the same located virtual machine in the Amazon EC2 cloud platform.

In 2014, Yuval Yarom [7] proposed the first cross-kernel channel attacks method called Flush-Reload. It exploits the vulnerability of the shared memory of the Intel X86 processor system and obtains the Cache-line processed by the CPU by monitoring the memory. This type of attack mainly uses the L3 layer Cache to implement the attack, and does not need to share the execution core.

In 2015, Liu [8] and Irazoqu enabled Prime-Probe attacks to be applied to cross-kernel Cache Side-channel Attacks.

In 2016, Irazaqui [9] proposed the first Cache attack on the AMD platform that can cross CPU. The attack environment gradually changed from a single core to cross-core, from the microprocessor to the cloud environment.

Domestic research started later, most of the attacks were timing attacks, i.e. cache timing attack is used to attack date (block) encryption, the correlation of the algorithm in each round of encryption is analyzed, and the key space is reduced by combining the data of multiple sample tests. Or according to the principle of asymmetric cryptographic algorithm, combined with the method of timing attack through mathematical attack, such as RSA algorithm in the process of decryption, its modular exponentiation uses binary multiplication from left to right, the algorithm in the process of execution, the key bit When it was 1, extra multiplication or squaring is performed, resulting in more execution time than the key bit is 0, so that the attacker can infer the RSA private key by inferring the key one by one by detecting the time difference.

Fangyong Hou [10] introduced the latest development of Cache- based AES attack research, analyzed the realistic feasibility of the attack, and explained the countermeasures for countermeasures against the attack; Hao Yang [20] introduced the timing attack and its preventive measures.

In 2008, Gaoming Deng [5] proposed a timing attack method based on cache hits and failure times of S-box access to attack block cipher process, and analyzed the first two rounds of cryptographic operations in DES cryptographic operations through experiments. Using the relationship between cache hit caused by these two array operations and the index value, he reduced the space for DES key search (The key search space is reduced from $2^{56}$ to $2^{13}$) on a 2.53 GHz Celeron processor with 512 MB of RAM. The time characteristic leakage points caused by accessing the Cache in the first round and the last round of the AES encryption process are analyzed. Based on this, the method of Cache time characteristic channel attack for the last round of AES is proposed. The idea can reduce the AES key search space from $2^{128}$ to $2^{20}$.

Chen Sen Chai [6] in accordance with the principle of asymmetric cryptographic algorithms, combined mathematical methods with timing attacks, resolved 1024-bit RSA private key in about two hours from an RSA encryption service program of simulation software, the attack takes about 350,000 sample cases.

Xinjie Zhao studies on AES [21], Camellia [22], SMS4 [11], CLEFIA [12] and other block cipher algorithms.

# 2. Introduction

## 2.1 CPU Structure

Modern computers execute user-specified instructions and data in a central processing unit (CPU). efficiency has become one of the main goals of structural design gradually, adding several hardware microarchitectures, such as cache or branch target buffer (BTB), in order to provide better performance. In addition, modern microarchitecture embedded several processing units in the same processor, some of these processing units can even process threads simultaneously. Furthermore, the rise of today's multi-CPU computers, Multiple of these CPU slots are embedded in the same piece of hardware. All of these technological advancements have the same goal: to provide users with the best computing performance.

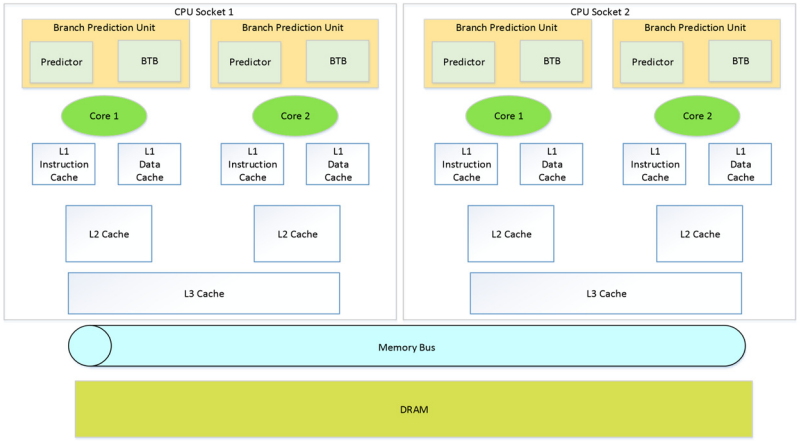


Figure 1. typical microarchitecture of modern processor

Typical microarchitecture design of common existing processors can be seen in Figure 1. This example shows a two-socket CPU with two cores per CPU. Each CPU core has its private L1 and L2 cache, and they share the last level of cache (LLC). In addition, each core has its own BTB that is responsible for predicting the outcome of the branch being executed. Communication between the cache and memory is done over the memory bus, which is also responsible for maintaining consistency between shared blocks across the CPU socket. Finally, the DRAM stores the necessary instructions and data needed for the program being executed.

While each component requires detail analysis, I focus on the cache (because of the legacy access content in the cache, access traces). meanwhile, some of the functions of the above structure will be described later in more detail if necessary.

## 2.2 Cache Structure

The hardware high speed Cache is a small memory placed between the DRAM and the CPU core, used to store data and instructions that may be reused very quickly. When the software needs read a specific memory block, the CPU first checks the cache level to find the date block. If found, the memory block is fetched from the cache and the access time is significantly faster. If a date block is not found in the cache level, the block is fetched from the DRAM at the expense of slower access time. These two situations are often referred as cache hits and cache misses respectively.

The main problem may be how fast the cache access compare with DRAM access. To this end, continuous timing access is performed on the L1 and L3 cached date blocks and the uncached memory blocks in the Intel i5-3320M. The L1 cache has an access time of probably 3 cycles because the L3 cache is probably 7 cycles and access to the main memory takes probably 25 cycles. Therefore, access to DRAM is probably three times slower than accessing the lowest cache level, which gives the concept of performance improvements provided by the cache level for software execution.

### 2.3 Inclusive Characteristics of Cache

An important item that cache designers need to consider, perhaps the most important character is whether they have inclusive, non-inclusive or exclusive cache. This has a serious impact on the cache-consistency protocol:

Include Cache: The include cache is the cache that any block that is required to exist in the upper cache (L1 or L2) must also exist in the LLC. Since LLC is shared, therefore several simplifications in maintaining cache consistency between the kernels. The inclusive property itself is responsible for maintaining consistency between shared memory blocks across the CPU core. A disadvantage of including a cache is that extra cache lines waste several copies of the same memory block.

Exclusive cache: Exclusive cache requires that the date block exists at only one cache level at one time. In contrast to inclusive caching, here the cache consistency protocol is implemented with a higher level of caching. However, they do not waste cache space to maintain multiple copies of the same block.

Non-contained cache: There is no requirement for such a cache, a block of date can exist in one or more cache levels at a time.

Whether or not the cache has inclusive characteristics can also be a key factor in implementing cache attacks. In fact, attacks such as Prime and Probe may only apply to inclusive caches, while other attacks such as Invalidate and Transfer doesn't need inclusive properties.

## 3. Cache Attacks Methods

Information leakage caused by shared Cache can be divided into two classes: information leakage caused by Cache compete and information leakage caused by data reuse. All Cache attacks methods are also based on this vulnerability.

### 3.1 Evict and Time Attack

The Evict and Time attack was proposed as a method to recover the full AES key [13]. In particular, this work demonstrates that key-related T-table access in an AES implementation can result in the leak of key. The method used can be described as:

- The attacker performs full AES encryption using known fixed plaintext and unknown fixed keys. After complete this step, all data used by AES encryption exists in the cache.
- The attacker evicts a specific set in the cache. Note that if the AES process uses this particular set, the data it uses will no longer exist in the cache.
- The attacker performs the same encryption using the same plaintext and key and measures the time to complete the encryption. To a large extent, the time it takes to perform encryption depends on whether the attacker evicted the set used by AES in step 1. If do, encryption will take longer because data must be fetched from memory. If it does not, the attacker guesses that AES encryption does not use the set of his evictions and may discard some key candidates.

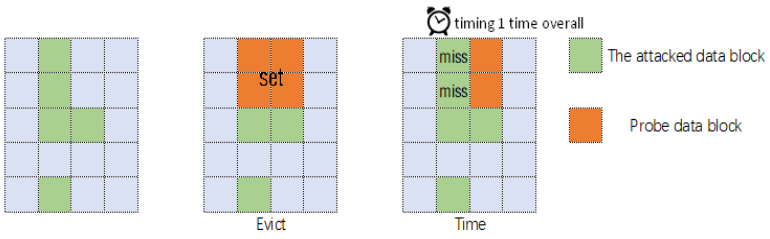


Figure 2. Evict and Time Attack

In Figure 2 the state described is represented graphically. The victim first using green cache block, the attacker expels two of them in the Evict process. When it is again clocked by the victim, the attacker see victim fetch two blocks from main memory, and infer that the victim used it to evict the collection. Please note that in this case, the attacker must use the same plaintext and encrypted key

record twice, which is a bit unrealistic. In addition, since the measurement is the overall encryption time (leakage information), although this attack method can judge the Cache compete of the entire process, it can only determine which interval it is, and it is difficult to judge the replacement of the specific location (block). And the current computer uses Cache multitier structure, resulting in the measurement of the substitution time is value of the interval rather than a certain value, so this method is not very practical.

### 3.2 Prime and Probe Attack

The Prime-Probe (PP) method was originally proposed by Osvik [30] for a stand-alone computing environment, after which PP attacks [14,15,16] is widely used by cryptographers. In a sense, the attack is similar to Evict and Time, but only needs to monitor its own memory blocks. This is a distinct advantage, because it's easier to implement and more meaningful. The main steps:

- Attackers fill L1 cache with its own dirty data.
- The attacker waits for the victim to execute the password process. the memory access will compete for some blocks occupied by the attacker in L1.
- Attackers access their own memory blocks and measure the time required to access them. If an attacker observes high access times for certain memory blocks, it means that the target software is exploiting the set of those blocks. Conversely, if an attacker observes a low access time, it means that the collection remains unchanged during the execution of the password process. Which could be used to restore the AES, RSA, or El Gamal keys.

The Prime and Probe steps can again be displayed graphically in figure 3. The attacker first fills the collection with the ready orange block of memory, then waiting for the victim execute the password process using the cache (the green block), and finally measures the load time of the orange block of memory's reaccess. In this case, it triggers two misses in the probe step, which means the victim use the appropriate set. Notice that the Prime and Probe attack display is more useful than the Evict and Time. because it doesn't measure the victim's progress. However, since it can only be applied to L1 caches, it is still not considered practical enough to be executed in a real-world scenario.

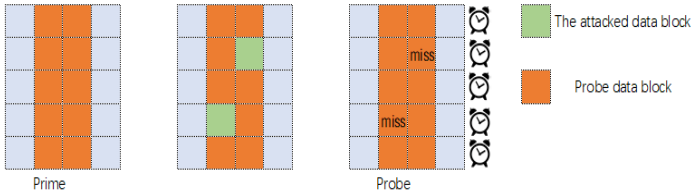


Figure 3. Prime and Probe Attack

### 3.3 Flush and Reload Attack

In 2013, some people proposed a special cache attacks idea based on the Intel processor's L3 cache architecture, and named this attack as Flush and Reload. This design utilizes the idea of shared cache among threads modern processors, the target thread is hijacked and attacked by Flush and Reload, as shown in figure 4, the system adopts copy-on-write (COW) scheme for the mapping of Shared page -- that is, the modified data is delayed to be written into memory, which is also a basis for FR attack to lead to information leakage.

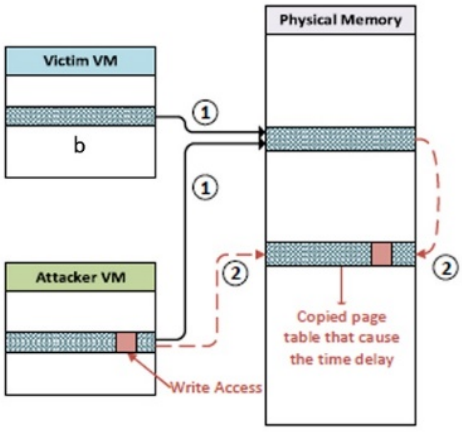


Figure 4. Copy-On-Write Scheme

As shown in Figure 5:

- Flush stage: At this stage, the attacker uses the *clflush* instruction to flush the orange monitored block out of the cache and ensure that it is retrieved from main memory the next time it needs to be accessed. The *clflush* instruction not only clears blocks of memory from the corresponding working core's cache level, but because of the LLC character of the inclusion, it refreshes all caches for all cores. This is an important point: if it only refreshes the corresponding core cache, then the attack is only effective if the attacker and victim's processes co-reside on the same core.
- Trigger state: At this stage, the attacker waits for the target to run a code snippet that may use a portion of the data block that has been monitored in the first stage of the memory block.
- Reload stage: At this stage, the attacker again reloaded and measuring the time required to reload the refresh step 1 out memory block. This can be done using a (user space accessible) *rdtsc* instruction that calculates the hardware cycles spent executing the process. Before reading the loop counter, you need to issue memory barrier instructions (*mfence* and *lfence*) to ensure that all load and store operations are completed before the memory block is read. The reloading time, an attacker can determine whether the victim has accessed the monitored memory block (in this case, the data block will appear in the cache) or whether the victim has not accessed the corresponding memory block (in this case, the data block will be taken out of memory).

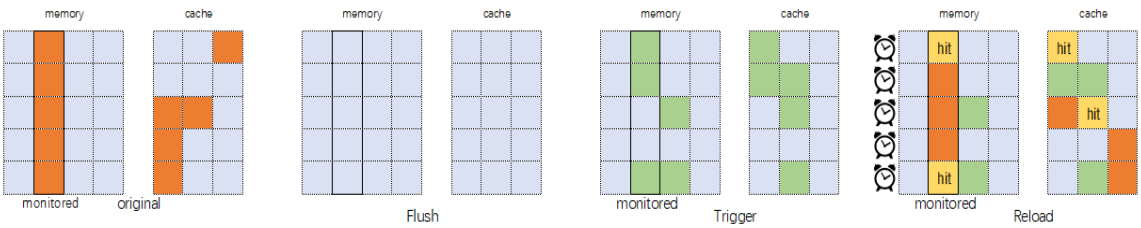


Figure 5. Flush and Reload Attack

Flush and Reload conditions : (1) Shared memory with victims: FR attack assumes that the attacker and victim at least share the target memory block in the system, that is, both access the same physical memory address.(2) CPU socket shared by victim and attacker: FR attack is only applicable if the attacker and victim coexist in the same CPU socket.(3)The inclusion LLC: Flush and Reload attack requires the inclusion character in LLC.(4) access refresh instruction: the refresh and reload attack of very specific instructions in the instruction set architecture (ISA) needs to be able to force the removal of memory blocks from the entire cache levels. In x86-64 systems, this is provided by the *clflush* instruction.

The timing differences between cache hits and cache misses make it easy for attackers to detect access rights. In fact, this is a big advantage of positioning LLC as a hidden channel. Fig 6 shows a

reload time is retrieved from memory blocks and memory LLC, respectively red and blue histograms. LLC access using FR technology typically requires probably 70 cycles, while memory access typically requires probably 200 cycles. Therefore, an attacker using FR technology can easily distinguish when an attacker uses a shared memory block.

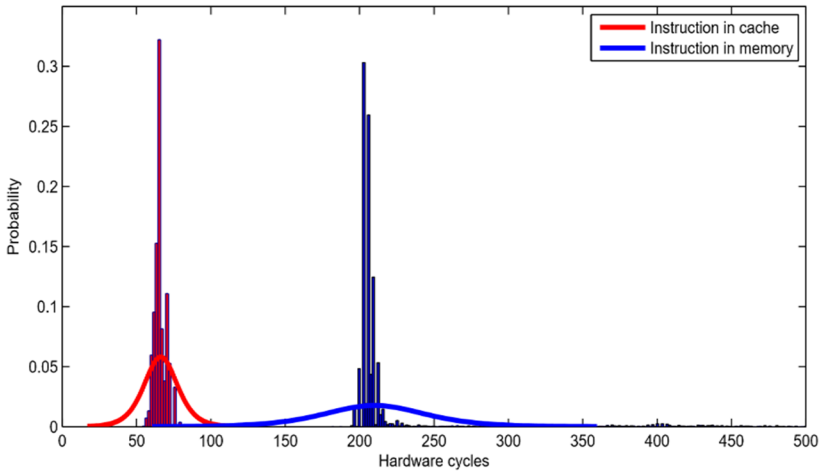


Figure 6. reload time in the hardware cycle when coexistence VMS use memory block b (red, LLC access) and when KVM is used on Intel XEON 2670 without using target memory block b (blue, memory access)

## 4. APPLY on CPU Vulnerability

An attacker can use the underlying shared resources of the Cache to construct a covert channel, bypass the logical isolation provided by the virtualized environment, and secretly steal the private information of other memory processes.

In modern computers, a monitoring bit is usually set in the processor between the kernel process and the user process to achieve isolation. Monitor bits can only be set when you enter kernel code (But it only maps the kernel to the address space of each process, so the memory map does not change when switching from the user process to the kernel.) However, we find that the optimization strategy of the modern computer to perform Speculative execution and out-of-order execution will leave the target we want to attack in the Cache, so we can fetch from the Cache by constructing a hidden channel.

Speculative execution: Its basic idea is to pre-determine the most likely position of the next instruction to be fetched in the local area, that is, it has a partial execution function in the fetching stage.

Out-of-order execution: It refers to the technology that the CPU allows multiple instructions to be sent to the corresponding circuit units in a sequence not specified by the program. In this way, according to the state of each circuit unit and the specific situation in which each instruction can be executed in advance, an instruction that can be executed in advance is immediately sent to the corresponding circuit.

The speculative can be executed as in nuclear before a judge (a) calculate the direction of the branch, carried out the most likely to judge the result in advance (read and calculations of the operation, but not for write operations), when the local area scope circuit unit does not have the use conflict, Out-of-order execution is understood to mean that unrelated instructions can be executed in parallel, because branch prediction to perform the read operation, put the goal we want to steal (memory block) in the Cache, this is the target. Introduction to Meltdown attack methods:

```
1    rcx = kernel address
2    rbx = probe array
3    retry:
4    mov al, byte [rcx]
5    shl rax, 0xc
6    jz retry
7    mov rbx,qword [rbx+rax]
```

Figure 7. Meltdown Attack core code

The attack process was shown in Figure 7:

Step 1: The contents of the memory location selected by the attacker are loaded into the register, which is not accessible to attacker.

Step 2: The temporary instruction accesses the cache-line based on the secret content of the register.

Step 3: The attacker uses Flush and Reload to determine which cache-line is being accessed, so the secret is stored in the selected memory location. By repeating these steps for different memory locations, an attacker can store kernel memory, including the entire physical memory.

Explanation: An attacker with user-level privileges attempts to access the kernel address in instruction 4 and the processor checks if the process has permission to access the address, so this instruction will trigger an exception, and the instruction and subsequent instructions will modify the register. It is discarded and the processor is returned to the instruction that can be executed normally. However, since the processor uses the out-of-order execution mode, while waiting for the processor to complete the execution of the instruction (before the permission check ends), the next two instructions have already been executed (although the final result will be discarded).

Multiply the data read from instruction 4 by 4096 (4KB), as the page size is 4kb, means writing data to the other address block, the result of instruction 5 as the index of the array probe probe_array (rbx [al * 4096]) to access and probe. Since the size of a memory page is 4KB, different data will cause different memory pages to be accessed and stored in the CPU cache. After that, the attacker can traverse the load rbx[al*4096] through the cache channel attack. Since the data is already in the cache at this time, the attacker will always traverse a data that is much smaller than other data, the kernel memory data being accessed is inferred from which memory pages have been accessed. To emphasize, the attacker's goal is to constantly probe the probe_array to get the data pointed to by the kernel address.

## 5. Problems with Realize Cache Attacks

### 5.1 Cache Usage Conflicts

All three attack methods use the timing method to measure the store or expel time. However, when the target process is executed, if the same memory block is loaded at the same time as the spy process, the Cache usage conflict may occur, and the recording time is imprecise. As shown in Figure 2.1, if the spy process and the target process are both in the same core of the same CPU (The Cache is distinguished time Shared), then the Cache conflict may occur during time collection. At this time, the attacker needs to be combined. The attacker encryption algorithm is used to write the scheduling time module of the spy program, but even this is difficult because the attacker process is continuously executed, and the next memory block used by the target process may be the last used memory block. There is a conflict with the spy timing process. If they are not in the same core, the Cache is shared at the same time.

### 5.2 Occupy Different Core Detection Problems

Because Cache Coherence problem was solved by LLC, just the spy and target process not in the same core, you can avoid cache conflicts, this requires multiple attempts to write the spy process into the Cache, and determine which process and target process not in the same core.

### 5.3 Method Limitations, Poor Popularity

The PP attack method is more suitable for L1. Because L1 has a small capacity, an attacker can use the virtual address of the data or code to determine its specific location in L1 (virtual addressing). If the PP method is applied to the LLC, LLC is private hash indexing mechanism. The mapping relation between the shards of the LLC and the physical memory address is determined by a non-public hash function. Even if the attacker can determine which cache-lines are included in a cache group, it is not known which part of the LLC these Cache-lines correspond to; For a large capacity LLC (typically greater than 2MB), the longer time for the entire measure LLC using Probe and Prime method.

The FR method is based on shared memory pages (memory blocks). In general, although the PP algorithm is more popular, the positioning is not precise, and the FR algorithm can obtain more precise information, but the limitation is large.

### 5.4 Building Hidden Channels is Unstable

First, computer vulnerability patch that has patched this vulnerability encryption process prevents multithreading at run time. so, the spy process is difficult to detect. Second, the Meltdown attack method has been retried multiple times (instructions 4-5), means the depth at of address enter into the Cache which instructions are unstable. After all, the speculative execution is not stable.

### 5.5 Other Aspects

Although the Cache attacks has related theoretical literature and simulation attacks in China, there are no real attacks on the PC. The main reason is the continuous improvement of the key algorithm and upgrade of the vulnerability patch. For example, OpenSSL is constantly patching and versioning. In the 0.9.8 a and later versions, the blinding technology is used to prevent against the timing attack, so that the security of the key bit is more than 1024 bits. On the other hand, the Cache attacks needs to measure time because of noise and the complexity of the computer, the time measurement has a certain error, so it is difficult to get better experimental results.

## 6. Conclusion

Based on micro-architecture attack since found largely ignored by cryptographers, mainly because of their lack of applicability in the real scene. In particular, they are subject to the following restrictions: ①Micro-architecture attacks only show success in core -private resources such as L1 cache and BTB. With the widespread use of multi-core systems, concerns applicability of microarchitecture attacks, the limitation of being in the same core as the victim seems to be a huge barrier. ② Microarchitecture attacks show success only in core-private resources (L1 cache and BTB) , L1 and L2 access differ only in a few cycles. BTB mistakes are also difficult to distinguish because their triggers have been optimized to a large extent. Therefore, core-private resources are less likely to resist the amount of noise typically observed in real-world scenarios. Until 2013, several methods of Cache attacks were made, and the vulnerability of the CPU architecture itself was taken care of by cryptographers because of its vulnerability. The existing security protection mechanism is difficult to detect and respond to such non-intrusive attacks. The use of Cache for Side-channel Attacks has become an important challenge to the privacy of users, and has received widespread attention from researchers at home and abroad [31-33], The Cache attacks are distinguished from other Side-channel Attacks. Among them, Flush and Reload, Prime-Probe, Evict and Time, which

are proposed by the combination of timing attacks, become the main attack methods, which are applied to different CPUs, and the leaked information is also obtained through specific analysis data.

## References

[1]. Menezes, AJ, Vanstone SA, Oorschot PV C. Handbook of Applied Cryptography [M]. Boca Raton: CRC Press, 2001.

[2]. Diffie W, Hellman ME. New Directions in Cryptography [J]. IEEE Transaction on Information Theory, V. IT - 22, 1976, (6): 644 - 654.

[3]. P. Kocher. Timing attacks on implementations of Diffie-Hellmann, RSA, DSS, and other systems. CRYPTO'96, 1996, LNCS 1109:104-113.

[4]. J. Kelsey, B. Schneier, D. Wagner, C. Hall. Side-channel cryptanalysis of product ciphers. Proc of 5th European Symposium on Research in Computer Security, pringer-Verlag, 1998, LNCS 1485: 97–110.

[5]. Gaoming Deng, STUDY ON CIPHER SIDE-CHANNEL ANALYSIS BASED ON CACHE TIMING CHARACTER, 2008.

[6]. Caisen Chen, Timing Attack Research of RSA Public Key Cryptography, 2008.

[7]. YAROM Y, FALKNER K. FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack [EB/OL]. Cryptology ePrint Archive, <http://eprint.iacr.org/> Report 2013 /448, 2013.

[8]. Fangfei Liu, Yuval Yarom, Qian Ge, Gernot Heiser, and Ruby B Lee. Last-level Cache 116 side-channel attacks are practical. In IEEE Symposium on Security and Privacy, pages 605–622, San Jose, CA, US, May 2015.

[9]. Gorka Irazoqui, Mehmet Sinan Inci, Thomas Eisenbarth, and Berk Sunar. Fine grain cross-VM attacks on Xen and VMware. In Proceedings of the 4th IEEE International Conference on Big Data and Cloud Computing, Sydney, Australia, 2014.

[10]. Fangyong Hou, Da-wu GU, Xiaoyong Li. Cache-Based Attacks against AES: Research Progress [J] Information Security and Communications Privacy, 2007, 7: 14-21.

[11]. Xinjie Zhao, Tao Wang, Yuanyuan Zheng. Cache timing attacks on SMS4 [J] Journal on Communications, 2010,31 (6): 89-98.

[12]. Xinjie Zhao, Shize Guo Wang Tao, Huiying Liu. Improved Cache trace driven attack on AES and CLEFIA [J] Journal on Communications, 2011 32 (8): 101-110.

[13]. Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache Attacks and Research: The Case of AES. In Topics in Cryptology-CT- RSA 2006: The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2005. Proceedings, pages 1-20, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[14]. Onur Aciicmez. Yet Another MicroArchitectural Attack: Exploiting I-Cache. In Proceedings of the 2007 ACM Workshop on Computer Security Architecture, 2007. [15].

[15]. Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, You, get o of My Cloud: Exploring Information Leakage in Third-party Compute Clouds. In Proceedings of the 16th ACM Conference on Computer and Communications Security, CCS '09, pages 199-212, 2009.

[16]. Yinqian Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. Cross-VM Side-channels and Their Use to Fetch Private Keys. In Proceedings of the 2012 ACM Conference on

Computer and Communications Security, CCS '12, pages 305 -316, New York, NY, USA, 2012.ACM.

[17]. Hongming Liu. RESEARCH AND IMPLEMENTATION OF COUNTERMEASURES AGAINST SIDE-CHANNEL ATTACKS FOR PUBLIC KEY [D]. Shanghai Jiaotong University, 2014.

[18]. Chunhui Sun. Research and Implementation of Side-channel Attack and Countermeasure [D]. Xidian University, 2012.

[19]. KOBLITZ N. Elliptic curve cryptosystems [A], Mathematies of Computation [C], 1987, vol. 48: 203-209.

[20]. Xi Yang. Timing Attack and Its Defense [J]. Communication Technology, 2008, 7: 185-188.

[21]. Xinjie Zhao, Tao Wang, Yuanyuan Zheng. Access Driven Cache Timing Attack Against AES [J] Journal of Software, 2011,22 (3): 572-591.

[22]. Xinjie Zhao, Tao Wang, Yuanyuan Zheng. Research on Access Driven Cache Timing Attacks Against Camellia [J] drive Journal of Computers, 2010, 33 (7): 1153-1164.

[23]. YAROM Y, FALKNER K. FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack [EB/OL]. Cryptology ePrint Archive, <http://eprint.iacr.org/> Report 2013 /448, 2013.

[24]. ZHANG Y, JULES A, REITER MK, et al. Cross-VM side-channels and their use to fetch private keys[C]// Proceedings of the 19th ACM Conference on Computer and Communication Security: October 2012, Raleigh, North Carolina, United States. New York, NY, USA: ACM, 2012: 305-316.

[25]. China Cryptography Society. China Cryptography Development Report 2008 [M]. Beijing: Publishing House of Electronics Industry, 2008.

[26]. PAGE D. Theoretical use of Cache memory as a cryptanalytic side-channel[R]. Technical Report CSTR-02-003, Department of Computer Science, University of Bristol, 2002.

[27]. Zhiguang Qin, " Introduction. " Cryptography Algorithm-Survey and Trends [J]. Computer Applications, 2004: 3- 4.

[28]. Mao W B. Modern Cryptography: Theory and Practice [M]. Prentice Hall, July 25, 2003.

[29]. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms [C]. s CRYPTO 1984, LNCS196, 1985: 10-18.

[30]. Osvik DA, Shamir A, Tromer E. Cache attacks and countermeasures: The case of AES//Proceedings of the 2006 the Cryptographers' Track at the RSA Conference on Topics in Cryptology. San Jose, USA, 2006: 1-20.

[31]. Ristenpart T, Tromer E, Shacham H, et al. Hey, you, get off of my cloud: Exploring information leakage in third-party compute clouds//Proceedings of the 16th ACM Conference on Computer and Communications Security. Chicago, USA, 2009:199-212.

[32]. Zhang Y, Juels A, Reiter MK, et al. Cross-VM side channels and their use to fetch private keys//Proceedings of the 2012 ACM Conference on Computer and Communications Security. Raleigh, USA, 2012: 305-316.

[33]. Liu F, Yarom Y, Ge Q, et al. Last-level Cache side-channel attacks are practical//Proceedings of the 2015 IEEE Symposium on Security and Privacy. San Jose, USA, 2015: 605-622.