Duplicating LLC based Side Channel Attacks on ARMv8 and A New Defense using xxx

Naiwei Liu UTSA
One University Circle
San Antonio, TX 78258
naiwei.liu@utsa.edu

xxx, xxx, Meng Yu
One University Circle
San Antonio, TX 78258
meng.yu@utsa.edu

ABSTRACT

In most modern commodity systems, the inspiration of design lies in which part we should trust in the computing processes. Based on this concern, popular feature-rich commodity operating systems separate a trusted computing base from other malicious or compromised parts. As a result, legacy applications run on untrusted systems, and a hypervisor or trusted hardware could keep the OS from accessing the memory of applications. On Armv8, TrustZone had been implemented on EL3 with the structure aiming at dividing the running of protected applications with others that are untrusted.

However, several people had successfully extract information from protected applications using side-channel attacks. In this paper we would have a discussion on side-channel attacks based on duplicating Last Level Cache(LLC) on ARMv8 platform. We will firstly try duplicating LLC-based side-channel attacks on ARMv8 platform, and then discuss on the ways for detecting these kinds of attack. A new defense strategy against LLC-based side-channel attacks will be introduced, with the evaluation on the performance on several tests later on. Given these results, we will find that using Perf (or others) to detect LLC-based side-channel attacks could be applicable.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Security

Keywords

Cloud Computing

1. INTRODUCTION

A significant number of systems has been designed working on shielding protected applications. They had used special trusted hardware, or set up hypervisors to keep the operating system from reading or writing application memory. As a result, even if the OS could be malicious, the trusted computing base would effectively protect applications' memory, making it safe for users to operate on these applications.

However, though the shielding computing structure could effectively protect most applications, some legacy code may still not be protected from malicious OS. As a result, the shielding system has to completely insulate the application from all types of adversarial action by the untrusted operating system. As the operating systems have to deal with resource management and provide services to the applications, the integrity of safety controlling could be a problem.

In the papers [], control flow integrity was discussed, with the ways to protect a system from Iago attacks. As these paper discussed, malicious OS could craft the output of system calls, acting like normal system calls. To carefully defend against Iago attacks, we could check control flow integrity, which contains checking process in every step of control flow, and make a decision before the system calls processing into application code. The security systems, which have control flow integrity check, would recognize the malicious system calls that tend to act like the normal ones. Those kind of systems are seen to be safe until recent years, when side-channel attack was frequently discussed and became an important threat to the security systems.

When setting up malicious side channel attacks, the attacker would firstly set up side channels to the victim systems, then watch the performance or other types of information retrieved from the victim system, and analyze the information leaked by side channels, mostly worked like a black box. The attacker always needs no internal understanding of the structure, instead, power consumption, timing information, or others like CPU temperature could provide extra information. These information collected by side channels can be exploited by the attacker to break in the system. As we have many tools for timing measurement, the most commonly used attacking way is to set up timing side channels.

For example, it is clear that the memory access time could be much difference if we load from main memory at every time. As a result, if someone who has no permission with looking into the memory access of an application, he might still be able to get the information from side channels based on the time difference. In recent years, more research had focused on the attack on last level cache, known as LLC. In a system with VMs or a cloud computing system, typically the LLC would be shared among the cores, making it risky and vulnerable to possible compromised cores with malicious applications and OS. However, unlike L1 cache, LLC cache has much slower access than L1 cache, making it hard for the attackers to set up side channels. One of the threat model was using FLUSH+RELOAD strategy[], which forces the attacker to

FLUSH LLC and thus achieving high resolution. It is proved that this strategy could work in some side channel attacks on x86 architecture.

We would implement LLC side channel attack on ARMv8 platform and take several tests to evaluate the performance of this type of attack. The type of FLUSH+RELOAD model will be used in the implementation, and we will watch the difference in time between the tests with FLUSH+RELOAD and without FLUSH+RELOAD. On the architecture of ARMv8, there are several types of difference between ARMv8 and x86. There could be different instructions for flushing cache, and the management of memory access could also be different. It is proved that adding some noise in cache side channel can be reduced in the situation when the attacker and the victim share the memory. As a result, FLUSH+RELOAD attack might be able to detected and disabled by some noise artificially added on. However, current research focused mainly on x86 architecture, so in this paper we would discuss the type of attack on ARMv8 architecture.

The major challenge of our research is the brand new structure of ARMv8. With the new architecture, new instruction set will affect on FLUSH operations. We have to implement with new in-line assembly code to ensure the operations on LLC are correct. On x86 architecture, as there are many ways to flush LLC, it could be easier to implement FLUSH operation. However, on ARMv8, there would be fewer banked registers and modes, FLUSH+RELOAD operations need to be re-designed. Another challenge introduced by this research is the discussion of overhead brought by the attackers. If we could see a wide difference in the performance with or without attackers, we might need to consider some effective ways both for the attacker and system defence.

In summary, our paper have made these contributions:

- We have implemented LLC side channel attack on ARMv8 architecture, and proved that it is a kind of attack that should be taken serious consideration.
- We have tested our attack on Juno r1 platform and collect the data of the performance with and without attacker, making attackers more effective on ARMv8 architecture.
- We have discussed the results and introduced a way to deal with the attackers. Experiments would be taken, proving that using XXX to defend against LLC side channel attack could be applicable.

Recent research [1],

2. RELATED WORK

LLC Convert Channels: Percival [] describes an L2 covert channel with a capacity of 100 KiB/s, but does not explain how the attack recovers the address mapping. Ristenpart et al.[32] experiment with L2 covert channels in a cloud environment, achieving a bandwidth of about 0.2 b/s. Xu et al. [] extend this attack, reporting an L2-based channel with a capacity of 233 b/s. By focusing on a small group of cache sets, rather than probing the whole cache, Wu et al. [] achieve a transfer rate of over 190 Kb/s. By accurately mapping the cache sets, our attack achieves a much higher bandwidth than prior work.

LLC Side Channel Attacks: Due to the low channel capacity, an LLC-based side channel typically only leaks course-grain information. For example, the attacks of Ristenpart et al. [32] leak information about co-residency, traffic rates and keystroke timing. Zhang et al. [46] use an L2 side channel to detect non-cooperating co-resident VMs. Our attack improves on this work by achieving a high granularity that enables leaking of cryptographic keys. Yarom and Falkner [] show that when attacker and victim share memory, e.g. shared libraries, the technique of Gullasch et al. [16] can achieve an efficient crossVM, cross-core, LLC attack. The same technique has been used in other scenarios []. Our attack removes the requirement for sharing memory, and is powerful enough to recover the key from the latest GnuPG crypto software which uses the more advanced 618 sliding window technique for modular exponentiation, which is impossible using FLUSH+RELOAD attacks.

In concurrent work Irazoqui et al. [23] describes the use of large pages for mounting a synchronous LLC PRIME+PROBE attack against the last round of AES. As mentioned above, [] had introduced cache flush attacks on x86.

FLUSH+RELOAD Attacks and Page Sharing: The FLUSH and RELOAD technique is a variant of PRIME+PROBE [] that relies on sharing pages between the spy and the victim processes. With shared pages, the spy can ensure that a specific memory line is evicted from the whole cache hierarchy. The spy uses this to monitor access to the memory line. The attack is a variation of the technique suggested by Gullasch et al. [], which include adaptations for use in multi-core and in virtualized environments.

A round of attack consists of three phases. During the first phase, the monitored memory line is flushed from the cache hierarchy. The spy, then, waits to allow the victim time to access the memory line before the third phase. In the third phase, the spy reloads the memory line, measuring the time to load it. If during the wait phase the victim accesses the memory line, the line will be available in the cache and the reload operation will take a short time. If, on the other hand, the victim has not accessed the memory line, the line will need to be brought from memory and the reload will take significantly longer.

The victim access can overlap the reload phase of the spy. In such a case, the victim access will not trigger a cache fill. Instead, the victim will use the cached data from the reload phase. Consequently, the spy will miss the access.

3. DUPLICATING LLC BASED SIDE-CHANNEL ATTACKS ON ARMV8

3.1 Design of the Attack

ARMv8 related instructions.

Process Structures and Implementations.
Please include critical codes here.

Experimental results.

4. DESIGN OF THE DEFENSE BASED ON XXX

5. CONCLUSIONS

6. ACKNOWLEDGMENTS

Meng's NSF xxx, xxx,xxx , SRO xxx, xxx, xxx.

7. REFERENCES

[1] Y. Xu, W. Cui, and M. Peinado. Controlled-channel attacks: Deterministic side channels for untrusted operating systems. In *Proceedings of the 36th IEEE Symposium on Security and*

Privacy (Oakland). Institute of Electrical and Electronics Engineers, May 2015.

APPENDIX

A. PROOF OF SECURITY OF THE COMMUNICATION PROTOCOL