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.

In //LLC attack on ARM, difficulty and originality //how to do it //eval. //intro the paper. Recent research [1],

2. RELATED WORK

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