The presentation transcript at the ASP-DAC 2020

Analyzing The Security of The Cache Side Channel Defences With Attack Graphs

Limin Wang, Ziyuan Zhu, Zhanpeng Wang and Dan Meng

P1: Good morning everyone, I am very glad to have the opportunity to share my work, the topic is how to analyze the security of the cache side channel defences.

P2: Let's start with the introduction.

P3:

1. The cache side channel leakage is a very serious issue in the information security field
2. There are four common cache attacks. They are Flush+Reload, Evict+Time, Prime+Probe, and Cache Collision.
3. Take flush+reload as an example
4. The attacker firstly carefully chooses memory addresses and flushes the cache lines mapped from these addresses.
5. Then, the attacker waits for the victim to read or write memory at the flushed addresses, and the accessed data will be re-cached.
6. Finally, the attacker tries to access the memory addresses chosen in step 1 and record the time of every access, if the access speed is fast, which means the current accessed memory address has been read or written by the victim in step 2, and has been cached.

P4:

1. Since it was reported in 2018, the hardware vulnerabilities like Meltdown and Spectre have been attracting a lot of interest
2. The Spectre attack firstly trains the branch predictor, and carefully make a wrong branch prediction that can be exploited by the attacker.
3. Then the processor will speculatively execute the attacker's codes.
4. During the speculative execution stage, the attacker can access the victim's data illegally and make them cached.

P5:

1. Hardware vulnerabilities can extend attack methods in some steps of the cache attacks, and make them more powerful
2. The traditional cache attacks need interaction with the victim.
3. But with Meltdown or Spectre, the attacker can access victim's data illegally and make them cached.
4. The risk of cache side channel attacks are increasing rapidly.

P6: Then we will introduce the motivation.

P7:

1. To defend against the cache side channel attacks and hardware vulnerabilities mentioned above, there are many defences been proposed.
2. It is obvious that, recently the scope of defences extends from software to hardware, that is because hardware defences can do a lot of things that software cannot do.
3. However, the hardware defences also have some problems
4. First, Hardware is hard to patch after been published. Second, It is also hard to design perfect defences.

5. Therefore It is necessary to analyze the security of the microarchitecture defences at the early stage of designing the processors.

P8: Then we will introduce our approach.

P9:

1. To analyze the security, in this paper, we proposed a model analysis method.
2. First, we will build a model for microarchitecture, and of course the defences on the microarchitecture will also be modeled.
3. Then we will develop the security specifications according to the cache attacks.
4. The model checker will check that whether the model satisfies the security specifications
5. If satisfies, the model will be considered to be secure. If not,the model checker will generate multiple counterexamples to prove that the model doesn't satisfy the security specifications
6. Then these abstract counterexamples will be used to generate a visualized attack graph
7. To achieve this,There are three challenges that we have to solve
8. The first challenge is that there are a lot of cache attacks and variants, whether there is a method that we can express these exploits conveniently
9. The second one is that the microarchitecture is complex, and how do we build a model that can be processed within the limited time
10. The last challenge is that the attack graph is a technology to express attacks in the network, how can we use it to express the cache attacks

P10: About Challenge 1, we will introduce the background, our approach, the detail of the method, and the advantages.

P11:

1. There are some features of exploits.
2. The cache attacks can be divided into several steps.
3. If the attack step is successful, the states of the system will become the states that the attacker expects.
4. it should be noted that the attacker can use different methods to make the system reach the insecure states.
5. And we find that the aim of the defences is to make the insecure states hard to reach.

P12:

Key Points: Fortunately, even though exploits are increasing rapidly,but the relevant insecure states do not.

1. Maybe we use different methods to achieve a successful attack, but we always focus on these insecure states.
2. Traditional methods usually model these attack steps and attack methods, therefore, they have to develop the specification for every exploit.
3. We model these attacks by using their insecure states, so that a specification can express a class of exploits.

P13:

1. When given a cache attack, we have to divide the attack into several steps.
2. And then we need to analyze the desired insecure states of each attack steps .
3. The sequence of the insecure states will be expressed by computation tree logics.

For example, the operation **EF** means Exist in the future, and the operation **U** means until.

The security specification here means that there should not exist a sequence in the future, and in this sequence, the states S2 holds *U*ntil S3 is true.

P14:

1. In this way, there will be two advantages
2. The first one is that a security specification can represent a class of exploits.
3. and the second one is that we can enumerate the attack paths that can reach these insecure states.

P15: As for Challenge 2, we will firstly introduce the background.

P16:

1. As we all known, microarchitecture is complex, and has many properties.
2. The states of the properties will change when the instruction executes.
3. And the states will also change due to the restrictions between different microarchitecture components.

P17:

1. The microarchitecture is really complex, so that model in the HDL level is **impractical** ,therefore, in this paper, we model the microarchitecture at the instruction level. our model can express the states change during the execution of the instructions.
2. It is noted that, our model has to satisfy the security specification, which means when the microarchitecture states change, there should not exist a sequence of insecure states.

P18:

1. The microarchitecture model consists of three sets, the S represents a set of microarchitecture states.
2. A microarchitecture state contains the current states of following properties. They are Attacker, Victim, instructions, and Microarchitecture components.
3. The "I" means the initial states,
4. And the R represents the transition relations. These relations describe the current states and next states

P19:

1. With the model and specifications mentioned above, we can use model checker to check whether our model satisfies our security specifications.
2. Here, we will give a simple example to explain how model checker works. Look at the picture, this is a state transition models, and the left part shows the formal definition of this model, Our specification is that there should not exists the path s1 to s2 to s3 in the future
3. And the model checker try to find the path in the model, and it finds one path that do not satisfy our specification, the whole path will be shown as a counterexample to prove that the model does not satisfy our specification

P20: Com back to our method, we build a model for microarchitecture, and develop the security specification. if the model doesn't satisfies the specification, the model checker will generate an counterexample to prove that the model does not satisfy the specification, which means our microarchitecture is insecure.

P21:

1. The abstract models contain and only contain the essential features, so that they can be used to express the microarchitecture conveniently.
2. On the other hand, the model has similar structures with attack graph so that the modeling method can facilitate the construction of attack graphs.

P22: The last challenge: How to generate the Attack Graph for cache attacks?

P23:

1. Previously, the attack graph is used to express the attack paths in the network. In another word, the attack graph can express which vulnerabilities did the attacker exploit in the network nodes. For example, the attack is at the node ip1, and he exploits the vulnerability4 to successfully attack the target.
2. The attack graph consists of three parts, they are preconditions, atomic attack, and postconditions. The preconditions mean that the states have to be true before the atomic attack performs. And the postconditions mean the states will be true after the atomic attack performs.

P24:

1. We have mentioned before that our microarchitecture model is a state transition model, which make the counterexample be a sequence of states.
2. And the counterexample has similar structures with attack graph.
3. So in our research, we will divide the counterexample into 3 parts, and transform them to the attack graph.

P25: We will take the current abstract instructions as the atomic attack, and the rest of the current microarchitecture state will be used as preconditions, and similarly, In addition to the abstract instructions ,the next state will be treated as the postconditions of current atomic attack.

P26:

1. The counterexamples have many repeated and similar results, therefore the attack graph generated have to be reduced.
2. there are 3 situations that will be reduced, the first one is shown in the figure (a), they have the similar structure, and the only difference is the atomic attack, after merging the repeated states, the result is shown in figure (b).
3. And figure (c) represents the situation that there are some repeated attack paths, the final one is logically equivalent.

P27: The counterexamples generated by model checker are abstract, and hard to analyze, the attack graph make them highly readable and easy to simplify.

P28: Next, we will take a simple experiment to explain how our method works.

P29:

1. We choose three different situations on the microarchitecture.
2. And we will analyze whether the microarchitecture is secure under Flush+reload or Evict +reload with Spectre.

P30:

1. The Model consists of microarchitecture states, initial states and transition relations.
2. We firstly introduce the states part.
3. These states are about the properties of the microarchitecture, as the table shows, we will model the cache, branch predictor and other properties of the processor. The attacker and the victim will also be modeled, the current executed abstract instructions is one of the properties of attacker and victim.
4. In this experiment, we choose these four abstract instructions.

P31: And then we will add the state transitions into the model, for example, for the load instruction, When the attacker execute the instruction, the state that attacker's data is in cache will change from false to true.

P32:

1. when there is a defence SP cache, SP cache can statically separates the cache for the victim and the attacker so that the attacker cannot evict victim's cache lines any more.
2. In the model of the microarchitecture with SP cache, we will remove the state transition about Evict strategy.

P33:

1. when there is a defence InvisiSpec, different from the traditional branch predictor, InvisiSpec load data into a new Speculative Buffer before committing the result until the speculative load is finally safe.
2. In the model of the microarchitecture with InvisiSpec, The state transition about the branch will be modified to the mechanisms the InvisiSpec describes.

P34:

1. Then we develop the specification for Flush+Reload and Evict+Reload with spectre. Both of them can be divided into 3 steps.
2. Look at the picture, during the attack, at a certain moment, the states will become s2, and the states keep, until at another moment, the states become S3.
3. Informally, the security specification can be described as this, which means there should not exist a sequence that S2 holds *U*ntil S3 is true in the *F*uture.

P35:

1. With these three microarchitecture model, and the security specification.
2. The results of the model checking are shown in the Table below.
3. When there is no defence, the model checker will find 247 counterexamples.
4. With the SP cache, the model checker can still find 81 counterexamples.
5. but when it is combined with InvisiSpec, there will be no counterexamples. Which means with InvisiSpec , the microarchitecture is secure enough under Flush reload or evict reload with Spectre.

P36:

1. Based on the counterexamples, we can generate the attack graphs.
2. The figure (a) is the attack graph of micro-architecture without any security designs.
3. The figure (b) is the attack graph of micro-architecture with SP Cache.
4. We can find that the SP Cache can only defend against the Evict strategy, and the flush operation can still bypass it.
5. We have to notice that, these two attack paths with a red circle, which means if when the attack starts, the secret data is not in cache, the evict strategy or flush operation will be not necessary, only Spectre and Reload steps can also finish the attack.

P37: Finally, we will make a conclusion.

P38:

1. Our method uses instruction abstract method so that we can model the microarchitecture conveniently.
2. Our method uses the sequence of insecure states to express the security specification so that one specification can express a class of the exploits.
3. we proposes a novel use of the attack graph technology to visualize the cache side-channel attack.
4. However, there still exists some limitations:
5. for example, we still cannot solve the state space explosion problem. which is a challenge in model checking.
6. and the attack that do not violate the security specifications will not be identified.