RISC-V SoC 환경에서 RoCC를 이용한 캐시 부채널 공격

황예원[O] 서태원 구건재

고려대학교 컴퓨터학과

hywp33@korea.ac.kr, suhtw@korea.ac.kr, gunjaekoo@korea.ac.kr

Cache Side-Channel Attacks Exploiting RoCC Interface on RISC-V SoC Platform

Yewon Hwang[O] Taeweon Suh Gunjae Koo

Department of Computer Science and Engineering, Korea University

요 약

System-on-Chip(SoC)은 많은 종류의 서드파티 IP를 활용하여 설계가 되고 있다. 서드파티 IP의 보안을 내부적으로 검증하는 것은 쉽지 않기 때문에 악의적인 공급자에 의한 하드웨어 트로이목마는 잠재적으로 심각한 보안 위협이 될 수 있다. 본 논문에서는 최근 많은 관심을 받고 있는 RISC-V 오픈소스 SoC 플랫폼 환경에서 RoCC 인터페이스를 활용한 캐시 부채널 공격 과정을 보여준다. 본 논문에서는 공격자가 가속기 IP를 제어하는 RoCC 명령어에 따라 동작하는 가속기에 악의적인 메모리 접근 로직을 설계하여 Flush+Reload 형태의 캐시 부채널 공격을 효율적으로 진행할 수 있음을 밝혔다. 본 논문에서 보여주는 공격 시나리오를 통하여, 본 논문은 오픈소스 ISA 및 명령어를 이용한 SoC 설계에 있어서 하드웨어 보안공격을 고려한 방어 방법 설계가 필요함을 제시하고 있다.

1. 서 론[1]

System-on-Chip(SoC)은 하나의 칩 내부에 여러 동작을 지원하는 다양한 종류의 모듈들을 포함하고 있다. SoC 설계 업체들은 제품의 시장 출시 시간을 단축하기 위해 사전에 설계된 서드파티 IP(third-party IP)를 외부 설계 업체로부터 공급받아 칩 내부에 포함시키고 있다. 이처럼 외부에서 제공받은 모듈은 자세한 내부 동작을 확인하기가 불가능하기 때문에 보안적으로 취약하다. 즉, 신뢰할 수 없는 공급자가 서드파티 IP에 설계하는 과정에서 악성코드 동작을 수행하는 하드웨어 모듈을 삽입하는 하드웨어 트로이목마(hardware trojan) 공격과 같은 보안 공격이 가능할 수 있다. IP모듈 공급자는 내부를 알 수 없는 형태로 해당 IP를 공급하기 때문에 하드웨어 트로이목마는 설계 과정 중에 발견하기가 매우 어렵다. 또한 하드웨어 트로이목마 공격은 소프트웨어 코드를 사용하지 않기 때문에 소프트웨어 패턴을 이용한 탐지도 거의 불가능하다.

본 논문은 현재 많은 관심을 받고 있는 오픈소스 ISA인 RISC-V 프로세서를 이용한 SoC 플랫폼에서 IP 모듈 및 가속기 동작 제어를 위해 사용하고 있는 RoCC 인터페이스를 이용한 캐시 부채널 공격을 제시한다. 기존 RISC-V ISA는 캐시 블록을 플러시하기 위한 별도의 명령어를 제공하지 않지만, RoCC 명령어로 제어되는 하드웨어 IP의 악의적인 동작을 통하여 보안 공격이 가능하다는 것을 본 논문에서 밝혀낸다.

* 이 성과는 2024년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임
(NRF-2021R1C1C1012172, NRF-2022R1A2C1011469)

2. 배 경

캐시 부채널 공격은 공격자가 캐시 블록에 저장되는 데이터 내용을 준비한 후에 피해자가 프로그램을 수행시키면 캐시 접근 패턴을 파악하여 피해자의 중요 데이터를 유출시킬 수 있는 보안 공격 방법이다. 캐시 부채널 공격은 많은 경우 캐시의 hit 과 miss 의 접근 시간의 차이를 이용하여 이루어진다. 이러한 캐시 부채널 공격은 다음과 같은 3 단계로 이루어진다[1].

① 공격자는 피해자가 실행하기 전에 캐시 블록에 저장되어 있는 데이터를 준비한다.

② 공격자는 피해자가 비밀 데이터를 이용하여 메모리 접근을 하여 캐시의 내용이 변경되기를 기다린다.

③ 공격자는 자신이 준비시킨 캐시 블록들에 접근하여 접근 타이밍을 측정하여 피해자의 비밀 데이터의 내용을 유추해낸다.

대표적인 캐시 부채널 공격 중 하나인 Flush+Reload 공격은 멀티 코어 구조에서 공격자와 피해자가 모두 접근할 수 있는 공유 캐시를 이용하여 캐시 부채널 공격을 수행한다[4]. 공격자는 ①단계에서 캐시 블록을 플러시 시킬 수 있는 캐시 제어 명령어(즉, x86 ISA 의 clflush 명령어)를 사용하여 캐시 블록을 invalid 시킨다. 이후 피해자가 메모리에 저장된 비밀 데이터에 접근하는 것을 기다린다(②단계). ③단계에서 공격자가 비밀 데이터를 유추하기 위해 ①단계에서 플러시시킨 캐시 블록들에 접근하여 캐시 접근 시간을 측정한다. 이 때 캐시 hit 과 miss 에 대해서 접근 시간이 다르게 측정되고, 이를 이용하여 비밀 데이터를 유추할 수 있다. 이처럼 캐시 부채널 공격은 메모리 계층 구조의 특성을 이용하여 빠른 시간 내에 효율적인 보안 공격이 가능한 특성을 가지고 있다. 그렇지만, 소프트웨어 형태의 보안 공격은 그 공격 패턴이나 소프트웨어 패턴을 탐지하여 보안 공격에 대한 방어가 가능하다[7].

3. RISC-V SoC 환경에서의 보안 공격

3.1. RISC-V SoC 플랫폼 구성

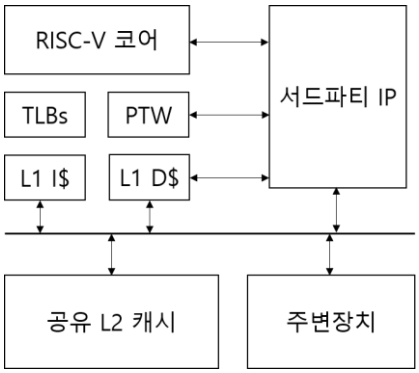

그림 1. RISC-V SoC Platform 구조

그림 1 은 보안 공격 실험에 사용한 RISC-V SoC 플랫폼 환경을 나타낸 것이다. 본 연구에서는 RISC-V SoC 개발에 사용되는 오픈소스 설계 플랫폼인 Chipyard 를 사용하여 SoC 개발 환경을 구성하였다[6]. Chipyard 환경에서는 RISC-V Rocket 코어를 CPU 로 이용하고 코어와 연결되는 가속기로 Gemmini systolic array 가속기를 사용하였다. Chipyard 에 포함되어 있는 가속기는 RISC-V 의 RoCC(Rocket custom coprocessor) 명령어 포맷을 통하여 동작이 가능하다. 본 연구에서는 Gemmini 가속기에 하드웨어 트로이목마가 삽입되어 있다고 가정한다.

RISC-V 코어는 L1 캐시와 TLB, page table walker(PTW)를 포함하고 있다. RISC-V 코어와 Gemmini 가속기는 interconnection network 를 통하여 L2 공유 캐시에 접근이 가능하다. 다음 표 1 은 이번 실험을 위해 구성된 RISC-V SoC 환경을 보여준다.

RISC-V core	Rocket core
L1 Cache	16KB, 64B/block
L2 Cache	512KB, 64B/block
Accelerator	Gemmini

표 1. 실험 구성

Rocket 코어는 연결된 가속기 IP 를 제어하기 위하여 RoCC 명령어를 사용한다. RoCC 명령어는 그림 2 와 같이 표준화된 RISC-V 명령어 포맷을 사용한다 [5]. RISC-V 코어는 RoCC 명령어를 통하여 가속기 내부의 레지스터 값을 업데이트하거나 CPU 의 레지스터 값들을 전달해 줄 수 있다. 가속기 내부의 레지스터 값 변경을 통하여 RISC-V 코어는 가속기의 동작을 간단한 명령어 실행을 통하여 제어할 수 있다.

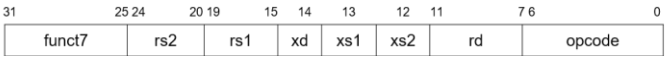

그림 2. RoCC 명령어 포맷

```
1 #define ROCC_INSTRUCTION_0_R_R(x, rs1, rs2, func7){
2       asm volatile(
3       ".insn r " STR(CAT(CUSTOM_, x)) ", "
4       STR(0x3) ", " STR(func7) ", x0, %0, %1”
5       :
6       : "r"(rs1), "r"(rs2));
7  }
```

그림 3. RISC-V명령어 매크로 코드

그림 3 은 RISC-V 코어가 RoCC 명령어 포맷을 이용하여 가속기로 제어 명령어를 전달하는 코드를 보여주고 있다. 즉, 위의 명령어는 RISC-V 코어 안의 rs1 과 rs2 로 지정된 레지스터 값을 읽어서 가속기 안의 레지스터로 저장하는 명령어를 보여주고 있다. 즉, 가속기 내부의 레지스터 값을 코어가 변경할 수 있으며 이를 통해 가속기의 동작 제어가 가능하다. 또한 가속기 내부의 레지스터 값을 코어가 읽어오는 것도 가능하기 때문에 가속기의 동작 상태를 모니터링 하는 것도 가능하다.

가속기가 활성화될 경우 가속기는 내부에 구현된 하드웨어 로직에 따라서 필요한 데이터를 메모리로부터 가지고 오며, 읽어온 데이터들은 설계된 구조에 따라 가속기 내부의 버퍼나 스트래치패드 메모리에 저장되게 된다. 그림 1 에서 보는 바와 같이 RISC-V 코어와 가속기는 L2 캐시를 공유하고 있으므로 가속기에서 읽어오게 되는 데이터는 L2 캐시에도 자동적으로 할당이 되게 된다. 가속기에서 설계된 동작이 끝날 경우 가속기는 내부 상태 레지스터에 값을 업데이트하게 되며 RISC-V 코어는 가속기 내부 레지스터 값을 읽어서 가속기의 상태를 파악하게 된다.

3.2. 가속기 IP 를 이용한 캐시 조작

이번 문단에서는 RISC-V SoC 플랫폼에서 RoCC 인터페이스로 연결되어 있는 가속기 IP 의 동작을 악의적으로 캐시 부채널 공격에 이용할 수 있는 상황을

설명한다. 가속기가 RoCC 명령어에 의해 활성화될 경우 가속기는 지정된 메모리 공간에 접근하게 된다. 그렇지만, RISC-V 코어는 가속기 내부에서의 메모리 접근 동작을 모니터링 할 수 없다. 악의적인 하드웨어 로직이 심어져 있는 가속기의 경우 지정된 데이터 접근 뿐만 아니라 임의의 데이터 접근이 가능하게 된다. 이 경우 가속기에서 수행되는 메모리 접근은 공유 L2 캐시를 통해 수행이 되므로 가속기의 임의적인 메모리 접근을 통해 L2 캐시의 내용을 조작할 수 있다. 비록 가속기가 코어와 직접적으로 연결되어 있는 L1 캐시를 조작하지는 않지만, 오픈소스 RISC-V SoC 설계 플랫폼인 Chipyard 는 inclusive 캐시 정책을 가지고 있기 때문에 L2 캐시의 내용 변경을 통하여 L1 캐시의 내용 변경이 가능하다. 즉, 가속기를 통하여 L2 캐시의 데이터를 플러시시키게 되면 inclusive 캐시 정책에 따라 L1 캐시에 있는 해당 데이터는 invalid 상태로 변하게 된다. 가속기의 동작이 활성화되기 전에는 해당되는 데이터는 L1 과 L2 캐시에 모두 존재한다. 가속기가 악의적인 메모리 접근을 통하여 해당 데이터가 저장되어 있는 L2 set 을 채우게 되면 해당 데이터는 L2 에서 evict 되게 된다. 동시에 해당 데이터는 L1 에서도 invalid 상태로 변하게 된다. 그러므로 해당 데이터에 대한 접근 시간이 더 길어지는 것을 볼 수 있다.

3.3. 악의적인 IP 를 이용한 Flush+Reload 공격

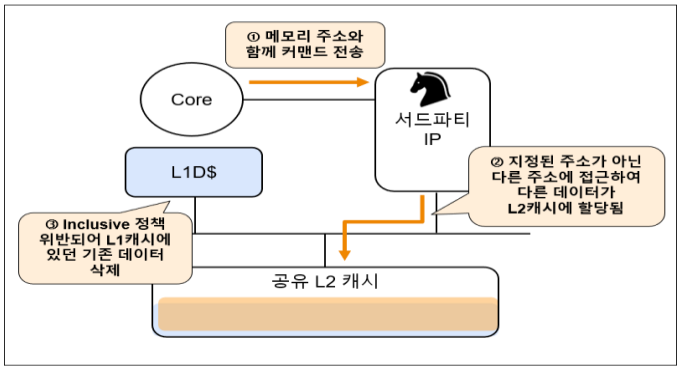

그림 4. 공격 시나리오

앞에서 설명한 대로 공격자는 RoCC 로 연결된 가속기 IP 에 하드웨어 트로이목마를 삽입하여 캐시 부채널 공격이 가능하다. 즉, 가속기가 활성화되었을 때에 L2 캐시를 의도하지 않은 데이터로 채움으로서 L1 캐시에 저장된 다수의 블록을 invalid 시키고 Flush+Reload 형태의 공격이 다음과 같이 가능해진다.

① 가속기에 심어진 트로이목마 로직이 가속기가 활성화되었을 때 지속적으로 메모리에 접근하여 L1 캐시에 있는 블록들을 invalid 시킨다(flush 단계).

② 공격자는 피해자가 비밀 데이터를 이용하여 메모리 접근을 하여 캐시의 내용이 변경되기를 기다린다.

③ 공격자는 invalid 상태로 변화된 캐시 블록들에 접근하여 접근 시간을 특정하여 피해자의 비밀 데이터 내용을 유추해낸다(reload 단계).

그림 5 는 보안 공격이 설계된 RISC-V SoC 플랫폼에서 공격자가 캐시 블록 인덱스를 이용해 접근했을 때의 접근시간을 측정한 것이다. 그림에서 보여지듯이 특정 인덱스에서만 낮은 접근 시간을 가지는 것을 볼 수 있다. 이와 같은 실험 결과를 통해 공격자는 RoCC 가속기에 심어진 트로이목마 로직을 통하여 효과적인 Flush+Reload 공격이 가능하다는 것을 알 수 있다.

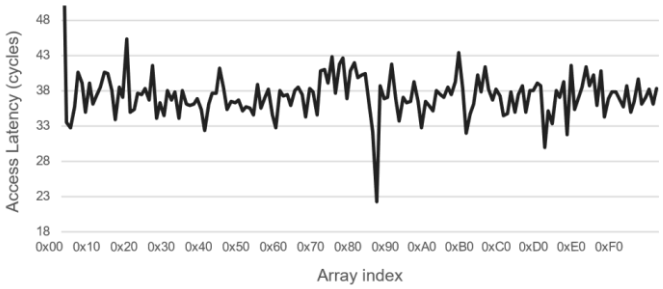

그림 5. 인덱스 별 접근 지연시간

4. 결 론

본 논문은 RoCC 인터페이스를 적용한 가속기 IP 를 이용하여 RISC-V SoC 플랫폼에서 캐시 부채널 공격이 가능한 시나리오를 제시하였다. 서드파티 IP 를 활발하게 이용하는 SoC 플랫폼에서 하드웨어 트로이목마 형태의 보안 공격은 매우 큰 보안 위협이 되고 있다. 본 논문에서는 현재 많은 관심을 받고 있는 오픈소스 ISA 및 SoC 개발 플랫폼을 이용하여 캐시 부채널 공격이 가능함을 보여줌으로써 현재의 오픈소스 SoC 플랫폼에서 이를 활용한 보안 공격을 방어하기 위한 기법들이 필요함을 밝혔다.

5. 참 고 문 헌

[1] Briongos, Samira, et al. "{RELOAD+ REFRESH}: Abusing Cache Replacement Policies to Perform Stealthy Cache Attacks." The 29th USENIX Security Symposium (USENIX Security 20). 2020.

[2] Asanovic, Krste, et al. "The rocket chip generator." EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17 4 (2016): 6-2.

[3] Pala, Davide. “Design and programming of a coprocessor for a RISC-V architecture.” Diss. Politecnico di Torino, 2017.

[4] Yarom, Yuval, and Katrina Falkner. "{FLUSH+ RELOAD}: A high resolution, low noise, l3 cache {Side-Channel} attack." The 23rd USENIX security symposium (USENIX security 14). 2014.

[5] Waterman, Andrew, et al. "The RISC-V instruction set manual." Volume I: User-Level ISA', version 2 (2014): 1-79.

[6] Amid, Alon, et al. "Chipyard: Integrated design, simulation, and implementation framework for custom socs." IEEE Micro 40.4 (2020): 10-21.

[7] Jin, Yier. "Introduction to hardware security." Electronics 4.4 (2015): 763-784.