스펙터 공격에 대한 방어 연구 동향

양지연 [1], 장진수 [2]
[1] 충남대학교 컴퓨터공학과 박사과정
[2] 충남대학교 컴퓨터공학과 교수

yjy1225@o.cnu.ac.kr, jisjang@cnu.ac.kr

A Survey on Defensive Measures Against Spectre Attacks

Ji-Yeon Yang[1], Jin-Soo Jang[2]
[1,2]Dept. of Computer Engineering, ChungNam National University

요 약

스펙터(Spectre) 공격은 CPU 최적화 기법인 예측 실행의 취약점을 악용한다. 이를 통해 프로그램의 정상적인 실행 흐름상 접근할 수 없는 데이터를 부채널을 통해 유출할 수 있다. 예측 실행 기법이 적용된 CPU 는 해당 취약점에 노출되어 있으며, 계속해서 새로운 변종 공격이 발견되고 있다. 지금까지 다양한 방어 기법들이 연구되어 왔고 새로 발견되는 변종 공격에 대응하기 위한 연구도 지속적으로 이루어지고 있다. 본 논문에서는 다양한 스펙터 방어 기법에 대한 연구 동향을 기술하고 향후 연구 방향을 제시한다.

1. 서론

스펙터 공격은 예측 실행 기술을 악용하여 정상적인 실행 경로로 접근할 수 없는 민감한 데이터를 유출하는 공격이다. 이 공격은 예측 실행 중의 결과가 마이크로아키텍처에 남은 흔적을 통해 부채널을 형성하여 데이터가 유출될 가능성을 야기한다. 스펙터 공격이 2017 년 처음 공개된 이후 이를 완화하기 위한 다양한 방어 기법이 연구 및 개발되었다.

본 논문에서는 스펙터 방어 기법에 대한 연구 동향을 살펴보고 이를 두 가지 기준으로 분류한다. (1) 방어 기법을 하드웨어 변경이 필요한 방식과 소프트웨어 변경이 필요한 방식으로 구분한다. (2) 각 방어 기법들을 방어 대상 스펙터 공격을 기준으로 분류한다. 분류를 통해 현재까지 제안된 방어기법들은 특정 부류의 스펙터 변이만을 방어하거나 모든 변이를 방어하기 위해서는 하드웨어 변경 또는 커널과 하이퍼바이저와 같이 도메인 분리가 가능한 조건이 성립될 경우에만 가능함을 확인하였다.

2. 스펙터 공격

2.1 예측 실행

예측 실행(Speculative Execution)은 CPU 성능 향상을 위해 구현된 최적화 기법이다. 이론적으로, CPU 는 분기나 명령어 간의 데이터 의존성이 해결되기 전까지 다음 명령어를 실행하지 않고 대기해야 한다. 이와 같은 대기 시간은 큰 성능 저하로 연결되며 예측 실행과 같은 기술을 사용하여 성능 저하를 낮출 수 있다. 프로세서는 다음에 수행할 명령어의 결과를 미리 예측하여 실행하고, 예측이 정확할 경우 대기 시간 없이 연속적으로 실행이 가능하다. 만약 예측이 잘못된 경우 수행된 연산 내용들을 폐기하여 예측 실행 이전 상태로 되돌린다. 따라서 아키텍처 상에서는 예측 실행했던 내용들을 알 수 없지만 실행의 흔적이 캐시와 같은 마이크로아키텍처에는 남게 되고 이는 캐시 부채널 공격을 통해 역추적 가능하다.

2.2 캐시 부채널 공격

캐시 부채널 공격은 음향, 전자파, 전력 등의 부채널 정보를 활용하는 부채널 공격의 일종으로, 캐시 접근 패턴을 분석하여 민감한 데이터를 추출한다. 이 공격은 시스템이 데이터를 캐시에 저장하고 접근하는 과정에서 발생하는 시간적 차이를 사용한다. 예를 들어, 특정 메모리 주소에 접근하는데 걸리는 시간이 상대적으로 짧은 경우를 캐시 히트로 판단하여 정보를 추론할 수 있다. 대표적인 캐시 부채널 공격 방식의 예로 Flush+Reload, Prime+Probe 공격 등을 들 수 있다.

<표 1> 방어 기법 분류

분류 기준	방어 기법	SafeSpec	InvisiSpec	ConTExT	SLH	Quarantine	DynPTA	SpecTaint	KASPER	SpecFuzz	Switchpoline	JumpSwitches
하드웨어 수정	Variant 1	√	√	√								
	Variant 2	√		√								
	Variant 4	√		√								
소프트웨어 수정	Variant 1				√	√	√	√	√	√		
	Variant 2					√	√				√	√
	Variant 4					√						

2.3 스펙터 공격 유형

스펙터 공격은 크게 4 가지 변이(Variant 1-4) 유형으로 나뉜다. Variant 3 은 멜트다운(Meltdown) 공격으로도 알려져 있으며 운영체제와 같은 특권 소프트웨어를 그보다 낮은 권한을 바탕으로(예: 유저 권한) 공격가능하게 한다. 본 연구에서는 동일 권한을 가진 스펙터 공격에 초점을 맞추어 Variant 3 은 다루지 않는다. Variant 1 은 Spectre-PHT, Bounds Check Bypass(BCB)라고도 하며, 조건 분기 명령어를 예측하는 패턴 히스토리 테이블(PHT)을 공격에 활용한다. Variant 2 는 Spectre-BTB, Branch Target Injection(BTI)로 알려져 있으며, 잘못된 예측을 통해 분기 타겟 버퍼(BTB)에 악의적인 주소를 주입하여 실행 흐름을 조작한다. 마지막으로 Variant 4 는 Spectre-STL, Speculative Store Bypass 라고 불리며, 쓰기 명령(Store)과 읽기 명령(Load) 사이의 의존성과 CPU 의 비순차 실행을 악용하여 임의 메모리 내의 비밀을 유출할 수 있다.

3. 스펙터 방어 연구

스펙터 공격에 대한 방어기법은 크게 하드웨어 변경이 필요한 방식과 소프트웨어 기반의 방식으로 나눌 수 있다. 본 논문에서는 기존 방어 연구를 두 가지의 방어 접근 방식으로 분류한다. 추가로, 적용 가능한 공격 유형별로 방어기법을 분류한다.

3.1 방어기법에 따른 분류

하드웨어 변경이 필요한 방어 기법에는 InvisiSpec, SafeSpec, ConTExT 등이 있다. InvisiSpec[1]은 데이터 캐시 계층에서 하드웨어 예측 공격을 막는다. 예측 실행되는 로드가 캐시에 직접 저장되지 않고 예측 버퍼를 사용하여 저장되며, 예측이 올바른 경우 버퍼의 내용이 캐시에 로드된다. SafeSpec[2]은 예측 실행을 위한 섀도우 하드웨어를 제안한다. 예측이 잘못된 경우 마이크로아키텍처 상태 변경을 취소할 수 있다. ConTExT[3]는 메모리와 레지스터의 민감한 데이터를 보호할 수 있는 기술로, 데이터가 마이크로아키텍처 상태로 누출되는 경우에만 일시적 실행(Transient Execution)을 중단하여 데이터 유출을 방지한다. 하지만 이와 같은 접근 방식들에는 하드웨어의 변경이 필요하거나 최신 아키텍처에만 적용 가능하다는 단점이 존재한다.

소프트웨어 기반의 방식 중 하나인 SLH(Speculative Load Hardening)는 예측 실행일 경우 로드된 데이터를 오염시키는 코드를 삽입하여 중요데이터 유출을 방지하는 LLVM 컴파일러 수준의 방어기법이다. Quarantine[4]은 커널, 하이퍼바이저와 같이 서로 다른 보안 도메인을 물리적으로 격리하여 마이크로아키텍처 수준의 리소스를 서로 공유하지 못하게 강제함으로써 알려진 스펙터 공격의 변이들을 대부분 방어한다. 하지만 동일한 권한을 바탕으로 실행되는 소프트웨어에는 적용하기 어렵다는 단점이 있다. DynPTA[5]는 정적 분석과 동적 데이터 흐름 추적을 결합한 방식으로, 주석(Annotation) 처리된 데이터의 하위 집합이 메모리에서 암호화된 상태로 유지되도록 구현된 기술이다. 이를 통해 예측 실행으로 인해 민감한 데이터가 유출되어도, 캐시에 암호화된 형태로 로드되기 때문에 데이터의 기밀성이 항상 유지된다. 위와 같은 소프트웨어 기반의 접근 방식은 구현 난이도 및 복잡성, 그리고 하위 호환성 측면에서 하드웨어 기반의 방식에 비해 장점이 있지만, 성능 저하가 크다는 단점이 있다.

이러한 소프트웨어 기반 방식의 성능 저하를 최소

화하기 위해 스펙터 가젯을 탐지하는 연구들이 있다. 스펙터 가젯은 공격에 사용되는 작은 코드 조각을 의미하며, 공격자가 스펙터 취약점을 악용한 공격을 수행할 때 사용된다. SpecTaint[6]는 동적 오염 분석 기법을 사용한 스펙터 가젯 탐지 기술로, 공격 대상 프로그램의 바이너리 수준에서 가젯을 식별하는 데 초점을 둔다. KASPER[7] 역시 오염 분석을 활용한 동적 분석 기반 접근 방식의 가젯 탐지툴이며, 공격에 대한 정확한 모델링을 통해 보다 넓은 범위의 가젯을 탐지할 수 있다. SpecFuzz[8]는 소프트웨어 퍼징 기술을 활용하여 스펙터 취약점에 대한 동적 검증을 가능하게 하는 도구이다. 이와 같은 기술들은 스펙터 공격에 악용되는 특정 코드 패턴을 식별함으로써 성능 저하의 요인이 되는 방어 코드 삽입을 최소화할 수 있다.

3.2 공격 유형별 분류

Variant 1 은 스펙터 공격 중 가장 활발하게 방어 연구가 진행된 대표적인 유형이다. 앞서 언급된 SLH, DynPTA, KASPER, SpecFuzz 등은 모두 스펙터 Variant 1 에 적용되는 방어기법이다. SpecFuzz 는 Variant 2, 4 에 확장이 가능하다. DynPTA 는 Variant 2 에도 적용 가능하며, 그 외에도 Switchpoline[8], JumpSwitches[9]와 같은 기술이 스펙터 Variant 2 를 방어하는데 사용된다. 두 가지 기법 모두 Variant 2 에서 사용되는 간접 분기를 직접 분기로 교체하여 공격 가능성을 최소화한다. Switchpoline 은 ARM 에, JumpSwitches 는 x86 아키텍처에서 각각 연구되었다. 마지막으로 Variant 4 는 SSBB 메모리 장벽을 삽입하거나 Quarantine 기법 적용 등을 통해 방어 가능하다. 또한 저장 및 로드 명령어의 비순차 실행을 방지하기 위한 시스템 레지스터인 SSBS(Speculative Store Bypass Safe)도 최신 ARM 기반 디바이스에서 사용 가능하다.

4. 결론

스펙터 공격은 예측 실행의 취약점을 악용하여 프로그램의 정상적인 실행 경로로는 접근할 수 없는 임의의 데이터를 유출하는 공격이다. 스펙터 공격이 공개된 이후, 다양한 방어기법에 대한 연구가 진행되어 왔다. 본 논문에서는 스펙터 공격 방어를 위해 제안된 몇 가지 방어기법들을 하드웨어/소프트웨어 기반 방어 방식별, 적용 가능한 공격 유형별로 분류하였다. 분류를 통해 지금까지 제안된 대부분의 방어기법들은 특정 공격 유형, 특정 아키텍처 또는 버전에 한정적으로 적용 가능함을 확인하였다. 향후 스펙터 공격 유형, 아키텍처 등에 영향을 받지 않는 범용 보안기술의 연구가 필요하다.

Acknowlegment

본 연구는 정보통신기획평가원의 지원(No.2022-0-01200, No.2021-0-00724, No.2020-0-01840)과 한국연구재단의 지원(RS-2023-00240697)을 받아 수행된 연구임.

참고문헌

[1] YAN, Mengjia, et al. Invisispec: Making speculative execution invisible in the cache hierarchy. In: *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2018. p. 428-441.

[2] KHASAWNEH, Khaled N., et al. Safespec: Banishing the spectre of a meltdown with leakage-free speculation. In: *2019 56th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2019. p. 1-6.

[3] SCHWARZ, Michael, et al. Context: A generic approach for mitigating spectre. In: *Network and Distributed System Security Symposium 2020*. 2020.

[4] HERTOGH, Mathé, et al. Quarantine: Mitigating Transient Execution Attacks with Physical Domain Isolation. In: *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*. 2023. p. 207-221.

[5] PALIT, Tapti, et al. Dynpta: Combining static and dynamic analysis for practical selective data protection. In: *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021. p. 1919-1937.

[6] QI, Zhenxiao, et al. SpecTaint: Speculative Taint Analysis for Discovering Spectre Gadgets. In: *NDSS*. 2021.

[7] JOHANNESMEYER, Brian, et al. Kasper: Scanning for Generalized Transient Execution Gadgets in the Linux Kernel. In: *NDSS*. 2022. p. 12.

[8] OLEKSENKO, Oleksii, et al. {SpecFuzz}: Bringing spectre-type vulnerabilities to the surface. In: *29th USENIX Security Symposium (USENIX Security 20)*. 2020. p. 1481-1498.