나를 합격시킨 토픽 - Meltdown & Spectre

오민석
(min-oh@korea.ac.kr)

Meltdown & Spectre vulnerability	
Concept	**(Meltdown 의 정의)** - 사용자 모드에서 커널모드의 데이터를 획득 가능한 인텔 CPU 마이크로아키텍처 관련한 취약점(rogue data cache load (CVE-2017-5754) **(Specture 의 정의)** - 서로 다른 응용 프로그램간에 보호되어야 할 데이터를 획득 가능한 CPU 아키텍처에 관련한 취약점(bounds check bypass (CVE-2017-5753), branch target injection (CVE-2017-5715))
KeyWord	**speculative execution, branch prediction, OoOE(Out of Order Execution), CPU 아키텍처 취약점, CPU 마이크로아키텍처 취약점**

치명적인 CPU 설계 결함의 발견

CPU 게이트로 불리는 이 사건은 2018 년 1 월 3 일 구글에서 최초로 발표하였다. 구글 프로젝트 제로의 얀 호른 수석연구원과 오스트리아 그라츠 공과대학, 그리고 업계 보안전문가들이 처음 발견했다. 흔한 보안 이슈가 아니라 게이트 소리를 듣는 대사건이었다. 특히 인텔 CPU 의 경우, 현역으로 사용 가능한 대부분의 제품이 버그의 위험에 노출되어 50 년 역사의 인텔 창업 이래 최악의 위기라는 우려도 나오는 상황이었다. 발견된 취약점 중 스펙터는 인텔을 포함하여 AMD, ARM 등 모든 CPU 에 존재하는 취약점이고, 멜트다운은 인텔 및 일부 ARM CPU 에 나타나지만 AMD 는 영향을 받지 않는 취약점이다. 이중 멜트다운은 CPU 보안을 위한 기본 구조인 Dual Mode 를 우회하는 취약점으로 매우 심각한 취약점으로 평가 받고 있다.

이번에 공개된 Meltdown & Spectre 에 관련된 취약점은 아래와 같은 3 가지 이다.

Variant 1: bounds check bypass (CVE-2017-5753)
Variant 2: branch target injection (CVE-2017-5715)
Variant 3: rogue data cache load (CVE-2017-5754)

이 중 Variant 1, 2 는 Spectre 취약점이며 Variant 3 은 Meltdown 취약점이다.

취약점 분석

Variant 1: bounds check bypass (CVE-2017-5753)

- 해당 취약점은 CPU 의 추측에 의한 코드가 미리 실행되는 최적화 기술(Speculative execution)의 맹점을 이용, 캐시 된 상태를 추론하는 방법으로 비인가된 메모리 영역의 값을 유추한다.
예를 들어, 아래 '그림 1 의 취약한 코드 예시' 같은 배열의 크기를 넘어서는 인덱스를 참조하려는 경우 조건에 의해 논리적으로는 해당 메모리 공간은 접근되지 않는다. 그러나 표현된 코드의 의도와는 다르게, 최신 프로세서는 빠른 실행을 위해 조건 검사 시점에서 병렬 프로세싱으로 조건문 내부의 코드를 미리 실행하여 준비해두고, 만약 조건이 맞지 않을 경우 미리 실행된 상태를 버리는 방식으로 동작하게 된다(파이프라인의 제어 헤저드를 해소하기 위한 방안). 다만 이 과정에서 프로세서는 미리 실행된 상태만 버릴 뿐, 빠른 메모리 재 접근을 위한 캐시 상태는 그대로 두게 되는데, 만약 악의적인 목적으로 배열의 인덱스를 비인가 메모리 영역으로 참조하게끔 구성한다면 비록 코드는 실행되지 않더라도 비인가된 메모리 영역이 캐시 상태에 남게 된다. 이 때 본 취약점의 원리는 메모리 접근 명령어를 실행하는 데에 걸리는 시간을 측정하여 대상 주소가 캐시 된 상태인지 추론하는 시나리오를 이용하여 지정한 비인가 메모리 영역의 값을 추론한다.

```
/*********************************************************************
 Victim code.
 *********************************************************************/
unsigned int array1_size = 16;
uint8_t unused1[64];
uint8_t array1[160] = { 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16 };
uint8_t unused2[64];
uint8_t array2[256 * 512];

char *secret = "The Magic Words are Squeamish Ossifrage.";

uint8_t temp = 0; /* Used so compiler won't optimize out victim_function() */

void victim_function(size_t x) {
  if (x < array1_size) {
    temp &= array2[array1[x] * 512];
  }
}
```

그림 1 취약한 코드 예시

Variant 2: branch target injection (CVE-2017-5715)

- 해당 취약점은 프로세서 예측 실행 기능 중 간접 분기예측(branch prediction)
 이용한다. 이 예측기는 간접 분기문을 해석하기 전 실행 된 분기문의 주소와 목적지 주소로의 매핑을 유지해주는 프로세서 분기 대상 버퍼(BTB)를 사용해 간접 분기문의 목적지 주소를 예측 실행한다. 이 때 동일한 프로세서에서 실행되는 프로세스들은 BTB 가 공유되기 때문에 BTB 를 사용한 예측 실행 시 프로세스들 간 영향이 발생 될 수 있다. 공격자는 자기 프로세스 간접 분기문의 목적지 주소를 희생자 프로세스 내부 가젯 주소로 분기하도록 변경하고 실행시켜 분기 예측 실패를 통해 프로세서의 BTB 를 수정시키게 된다. 이렇게 수정 된 BTB 는 공격자 프로세스

간접 분기문과 동일한 위치에 있는 희생자 프로세스 간접 분기문이 예측 실행 될 때 영향을 주게 된다. 다시 말하면 공격자 프로세스 간접 분기문에서 예측 실패 시 캐시는 버려지지만 BTB 에 미쳤던 영향은 제거되지 않기 때문에 이런 문제가 발생하는 것이다. 그 후, 희생자 프로세스 간접 분기문이 예측 실행 될 때 BTB 에 저장 된 목적지 주소는 가젯 주소이기 때문에 실제 목적지 코드가 아닌 가젯 코드를 예측 실행하게 된다. 예측 실행 된 가젯 코드는 공격자가 원하는 메모리 값을 캐시로 가져오게 되며, 그 이후엔 Variant 1 에서 사용 된 기술과 유사한 CLFLUSH 명령어를 사용하는 Flush+Reload 기술로 공격자가 원하는 위치의 희생자 프로세스 메모리 값을 알 수 있게 되는 것이다. 결국 Variant1,2 를 이용하게 되면 하나의 응용프로그램에서 다른 응용프로그램의 데이터를 획득 가능한 공격을 수행하는 것이 가능하다.

Variant 3: rogue data cache load (CVE-2017-5754)

- 멜트다운은 인텔 CPU microarchitecture 을 타겟으로 하는 공격으로서, 비순차적인 실행 방법(OoOE)을 이용하는 사용자의 커널 메모리를 유출시킬 수 있다. 공격자는 멜트다운 취약점을 이용해 프로세서에 있는 권한 상승 취약점을 공격한다. CPU 의 예측 실행 기능을 이용하면 공격자가 메모리 보호를 우회할 수 있기 때문이다. 해당 취약점을 이용하면 사용자 공간에서 커널 메모리에 접근하도록 허용된다. 이는 즉 사용자가 시스템 내부에 존재하는 보안 매커니즘(심지어 커널에 포함되어 있는 내용)과 관련된 다양한 코드 등에 접근할 수 있도록 허용하여 다양한 정보들이 유출될 가능성이 크다. 이러한 멜트다운 공격은 CPU 의 안정성과 Security 를 위한 OS 의 기본 구조인 Dual Mode 에 대한 우회가 가능한 취약점으로 반드시 대응 해야할 취약점이다.

취약점 대응

취약점	해결법	해당되는 CPU 벤더
Meltdown Variant 3 (Rogue Data Cache Load)	KPTI (Kernel Page Table Isolation) KPTI 가 필요한 CPU 일 경우 조건부로 커널 시스템 콜 핸들러가 사용될 수 있도록 수정	Intel (MS 는 AMD 에도 패치를 시행하였음)
Spectre Variant 1 (Bounds Check Bypass)	LFENCE instruction 인텔과 AMD 는 Spectre Variant 1 을 막기 위한 조치로 LFENCE 명령어의 사용을 권장. LFENCE 명령 다음의 명령어들이 CPU 로 로드 될 수 있지만, LFENCE 명령이 실행되기전까지는 실행되지 않도록 수정	Intel, AMD
Spectre Variant 2 (Branch Target	취약점에 노출된 CPU 의 마이크로코드 업데이트를 통해서 아래의 기능을 새롭게 추가	Intel

Injection)	- Enable Intel IBRS (Indirect Branch Restricted Speculation) - Enable Intel IBPB (Indirect Branch Predictor Barrier) - Enable Intel STIBP (Single Thread Indirect Branch Predictors) - Return trampoline(이것은 구글에서 제시한 수정 방법이며 인텔에서도 권장하고 있음)	