USB Keyboard Sniffing with TD

chpie275@gmail.com
http://beist.org/

0. Contents

1. Intro

대중적으로 사용되는 키보드 인터페이스는 PS/2, USB입니다. 데스크탑은 대부분 USB를 사용하고 있고 랩탑은 PS/2 방식과 USB 방식이 혼용되어 사용되고 있습니다. 그리고 최신의 랩탑 키보드들은 내부적으로 USB로 연결되어 있습니다. 본 문서에서는 USB 방식으로 작동되는 키보드를 대상으로 키보드 스니핑을 하는 방식에 대해 소개합니다.

USB 컨트롤러는 UHCI(Universal Host Controller Interface), OHCI(Open Host Controller Interface) 그리고 EHCI(Enhanced Host Controller Interface) 등이 있습니다. USB 장비가 시스템에 설치되었을 때, 빠른 처리를 요구하는 USB 장치의 경우에는 EHCI에 주로 연결되지만 키보드 같은 legacy 장치는 UHCI에 연결됩니다.

이때, UHCI는 주기억장치의 Shared Memory를 통해서 운영체제와 통신을 하게 됩니다. 이 문서는 이 Shared Memory를 거치는 UHCI 패킷을 가로채서 사용자가 입력한 키로그 데이터를 볼 수 있는 방식을 설명하고, 이를 막을 수 있는 아이디어를 소개합니다. 이 기술은 패킷의 내용을 볼 수 있을 뿐 아니라, 조작도 가능하며, 운영체제에 패킷이 도달하기 전에 이루어지므로, USB 필터 드라이버들의 영향을 받지 않습니다.

2. Design

먼저 UHCI에 대해서 알아보겠습니다. 운영체제는 UHCI 와의 통신을 위하여 주기억장치의 일부분을 Shared Memory로 사용해야 합니다. Shared Memory에서 사용되는 UHCI 데이터들은 프레임 리스트 헤더와 큐, 연결-리스트로 이루어져 있는데, UHCI 문서에서 Spec을 제공합니다.

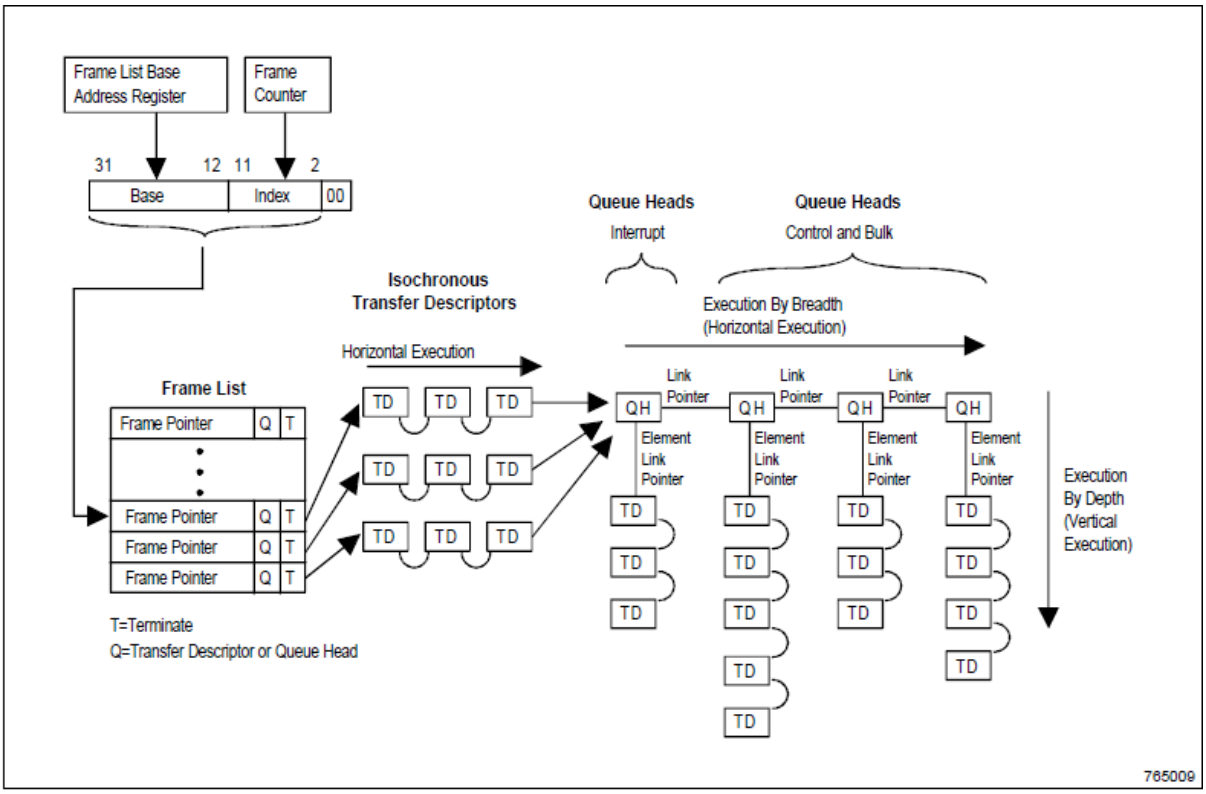


Figure 9. Sample Schedule Layout

<그림 1. Intel® Universal Host Controller Design Guide 에서 발췌>

TD라고 그려진 상자 하나하나가 패킷이며, Transfer Descriptor라고 부릅니다. Isochronous 전송의 경우 Frame Header에 연결되며, Interrupt, Control, Bulk 의 경우 Queue 에 삽입되어 순서를 기다립니다. Isochronous, Interrupt, Control, Bulk 에 대해서 자세히 알고 싶으신 분은, USB 스펙을 참고하시기 바랍니다. USB 키보드는 Interrupt 혹은 Bulk 방식을 사용하므로 우리는 Isochronous TD들은 신경쓰지않아도 됩니다.

Frame List에는 각 Frame에 대한 Entry가 존재하는데, 1 프레임에 대한 시간 x를 할당함으로써, x 시간이 흐른 뒤에는 다음 Entry를 처리합니다. Entry에 연결된 자료는 Isochronous TD가 되며, Isochronous 전송이 없을 시에는 바로 Interrupt Queue를 연결하고 있을 수 있습니다.

그림에 나타나는 모든 자료구조는 운영체제가 미리 준비해 두어야 하며, UHCI 는 미리 준비된 데이터들을 읽고 쓰면서 USB 장치들과 운영체제가 데이터를 주고 받을 수 있도록 합니다.

Frame List Base Address Register와 Frame Number Register (그림의 Frame Counter) 를 이용하여 현재 Frame에 맞는 Entry를 얻을 수 있습니다. Frame Number Register 는 x 시간이 흐른 뒤에 자동으로 1 씩 증가합니다.

각 Frame의 처리가 종료될 때 UHCI 는 시스템으로 인터럽트를 전송하는데, 각각의 Transfer

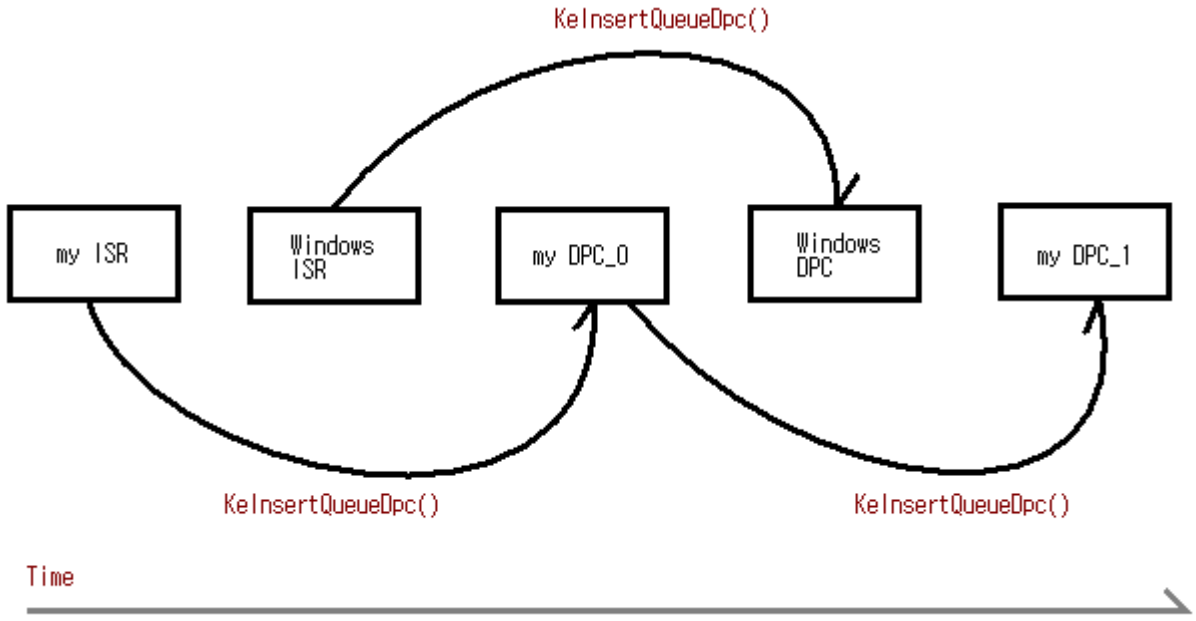
Descriptor의 처리가 종료되었을 때에도 인터럽트를 전송할 수 있습니다. 이는 UHCI의 설정에 따라 좌우되는데, Windows는 두 가지 인터럽트 소스를 모두 사용합니다.

Queue Head에 연결되어 있는 Transfer Descriptor 들은 UHCI가 이를 처리한 후에 Queue에서 제거됩니다. 이것은 중요한 부분인데, 키보드의 Transfer Descriptor가 처리된 후에는, 데이터들을 Queue에서 더 이상 찾아낼 수 없다는 뜻입니다.

Windows는 UHCI에서 인터럽트가 발생할 시에 UHCI와 공유하는 메모리를 스캔하는 것이 아니라, 내장된 리스트를 사용하여 Transfer Descriptor 중에 처리된 것들이 있는지 찾아내게 됩니다. 공유 메모리에 구축된 모든 자료구조들은 Physical Address를 이용하여 연결되어 있으며, Virtual Address로 구축된 자료구조를 Windows는 따로 가지고 있습니다.

즉, UHCI 인터럽트 핸들러를 후킹하여 Transfer Descriptor 가 처리된 시간을 알아내어도, 일반적인 방법으로 공유메모리를 스캔해서는 Transfer Descriptor 를 찾아낼 수 없습니다.

UHCI 는 기본적으로 x 시간을 주기로 작동하는 시간 기반 장치이며, CPU 에 비하면 현저하게 느린 속도입니다. 이를 이용하여 시간차 공격을 할 수 있는데, 이미 Transfer Descriptor 가 처리된 이후에는 찾아낼 수 없으므로, 처리되기 전의 Transfer Descriptor 주소를 미리 획득해야 합니다.



<그림 2 - Transfer Descriptor의 주소를 알아내기 위한 시간차 공격 개념도>

이 시간차 공격에서 DPC_1은 두 번 지연된 Deferred Procedure Call로써, Windows의 DPC 보다 나중에 수행이 됩니다. Windows DPC는 처리된 키보드 Transfer Descriptor에서 데이터를 키 입력 데이터를 추출한 후에, 다시 Frame List에 Transfer Descriptor를 Submit 하게 되는데, DPC_1이 수행되는 시점에서 대부분의 키보드 Transfer Descriptor는 아직 처리가 되지 않은 상태로 남게 됩니다. 이 말은, 일종의 레이스 컨디션을 이용하는 것이기 때문에 키보드 데이터를 못 가져오는 경우도 있을 수 있다는 말입니다.

100%의 확률을 가지진 않지만, 대부분의 Transfer Descriptor의 주소를 Queue를 조사하여 얻어낼 수 있으며, 이를 저장해 두었다가 다음 인터럽트 발생시 DPC_0에서 저장된 Transfer

Descriptor 들의 데이터를 추출합니다. 이미 실행된 Transfer Descriptor 들은 데이터의 크기를 나타내는 필드가 0이 아닌 값을 가지므로, 쉽게 판별할 수 있습니다. 또한 DPC_0은 Windows DPC 보다 먼저 수행되기 때문에, Windows 보다 앞서서 Transfer Descriptor 의 데이터를 조작하고 추출할 수 있게 됩니다.

사실 DPC_1에서 Transfer Descriptor 의 데이터를 추출해도 어떠한 키가 눌렸는지를 알 수 있습니다. 이는 Windows가 Transfer Descriptor를 submit 할 때 데이터 버퍼를 0으로 초기화 하지 않고 submit 하기 때문에 이전의 전송 내역이 버퍼에 남아있는 것입니다. 하지만 이 방식은 Windows 보다 먼저 Transfer Descriptor 를 처리하는 것이 아니기 때문에, 데이터를 보는 것만 가능하며 조작할 수는 없다는 단점이 있습니다.

3. Technique Evaluation

장점

- Windows 및 필터 드라이버 방식의 보안 프로그램보다 앞선 키 입력 데이터 선점
- 여러 키보드가 시스템에 설치되어 있어도 작동
- 하드웨어에 기반한 디자인이기 때문에 다른 운영체제에 환경에도 적용 용이함

단점

- 키 입력 데이터를 선점할 확률이 100%가 아님
- USB 추상화 계층의 혜택을 누릴 수 없음

이러한 특징을 이용하여 키보드 키로깅을 막을 수 있는 방법에 대해 간단하게 소개하겠습니다. 앞에서 다루었듯이 우리는 키 입력 데이터를 가장 먼저 선점할 수 있기 때문에, Transfer Descriptor에서 키 입력 데이터를 추출한 후 데이터를 0으로 만들어 버리면 됩니다. 하지만 이 역시도 100% 성공률을 보장하지 않기 때문에 보안 프로그램이 되기 위해서는 여러 가지 개선이 필요할 것으로 생각됩니다.

4. Reference

0. USB Keyboard Sniffer with TD, Proof of Concept code
   http://beist.org/research/public/usb_keyboard_sniffer/InpHID_withTD.zip

1. Intel® Universal Host Controller Interface Design Guide Revision 1.1
   http://download.intel.com/technology/usb/UHCI11D.pdf

2. Programming the Microsoft Windows Driver Model 2 (Book)

3. Bochs : Open Source IA-32 Emulation Project
   http://bochs.sourceforge.net/

4. USB Snoopy
   http://sourceforge.net/projects/usbsnoop/

5. Incle - USB Keyboard Hooking.pdf
   http://avinew.cafe24.com/