情報保護學會論文誌
第18卷 第4號, 2008. 8

키보드컨트롤러의 하드웨어 취약점에 대한 대응 방안

정 태 영, 임 강 빈†‡
순천향대학교

Countermeasures to the Vulnerability of the Keyboard Hardware

Tae-Young Jeong, Kang-Bin Yim†‡
Soonchunhyang University

요 약

본 논문은 문자열 기반 패스워드 인증에서 키보드 감시 문제를 유발하는 하드웨어 취약점에 대한 효과적인 대응방안을 제안한다. 악의적인 공격자가 해당 취약점을 이용하면 기존의 보안 소프트웨어가 실행되고 있는 상황에서도 키보드로부터 입력되는 사용자의 모든 문자열을 탈취할 수 있었으나 본 논문에서 제시하는 방안을 활용하면 공격자가 키보드 감시에 성공한다 하더라도 사용자의 입력 문자열을 온전히 탈취할 수 없다. 따라서 제안한 방안을 구현하여 적용하면 보다 안전한 인터넷 기반 금융거래가 가능해질 것으로 사료된다.

ABSTRACT

This paper proposes an effective countermeasure to an intrinsic hardware vulnerability of the keyboard controller that causes sniffing problem on the password authentication system based on the keyboard input string. Through the vulnerability, some possible attacker is able to snoop whole the password string input from the keyboard even when any of the existing keyboard protection software is running. However, it will be impossible for attackers to gather the exact password strings if the proposed policy is applied to the authentication system though they can sniff the keyboard hardware protocol. It is expected that people can use secure Internet commerce after implementing and applying the proposed policy to the real environment.

Keywords : scan codes, H/W vulnerability, sniffing, password expose, keyboard controller, internal memory

Ⅰ. 서 론

인터넷을 이용한 다양한 서비스가 출현하면서 오프라인에서만 이루어지던 많은 서비스들이 인터넷상에서 자연스럽게 이루어지고 있다. 특히, 몇 년 전만 하더라도 금융거래 서비스나 전자지불 서비스 등은 인터넷을 활용하는 경우가 흔치 않았으나 현재는 대부분의 사람들이 이러한 서비스를 인터넷을 통하여 수행하게 되었다. 이러한 현상은 앞으로는 더욱 더 보편화되어 향후 가상공간과 실제공간이 사상되는 상황에서는 거의 모든 사회적 작용이 인터넷을 통하여 이루어지게 될 것으로 전망된다. 따라서 정보보호의 문제는 단순히 개인의 정보를 생성, 저장, 이동에 대한 지원의 문제가 아닌 미래사회를 유지하는 중추가 될 것으로 전망된다. 상기와 같은 미래사회에서는 사용자나 장치에 대한 인증이 무엇보다 중요하며 이는 인증에 사용하는 도구 자체가 보안

접수일 : 2008년 5월 20일; 수정일 : 2008년 6월 11일;
채택일 : 2008년 7월 2일
† 주저자, yim@sch.ac.kr
‡ 교신저자, yim@sch.ac.kr

문제에 대하여 안전할 것이 보장되어야 한다.

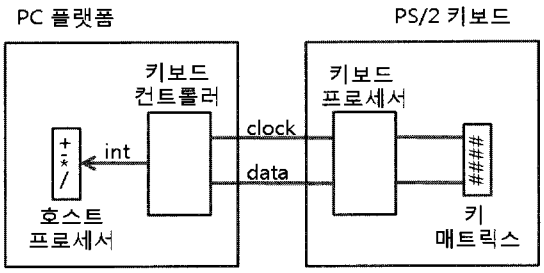
현재 사용 중인 대개의 인터넷 기반 서비스에서는 문자열 형태의 패스워드 인증이 이루어지며 이 과정에서 사용자 컴퓨터의 키보드를 통하여 패스워드를 수집하는 방식을 사용하고 있다. 한편, 과거 몇 년간 키보드의 안전성에 대한 의구심이 수시로 거론되어 왔으며 실제로 사용자 키보드의 입력을 감시하는 상용 소프트웨어들도 다수가 등장하였다[1]. 이에 따라 키보드 감시를 방지하기 위한 보안 소프트웨어가 주요사이트에서 제공되어 사용자에게 필수적으로 설치하여 운용하도록 요구하고 있다. 그러나 다양한 키보드 감시 방지기술에도 불구하고 컴퓨터의 PS/2 키보드 인터페이스 하드웨어로부터 기인하는 취약점에 대하여는 효율적으로 대응하지 못하였으며 보안 소프트웨어가 실행되는 상황에서도 사용자의 패스워드가 그대로 노출되는 문제가 제기되었다[2]. 2007년 7월 USB 키보드의 취약점이 지적됨에 따라 과거 정보통신부에서는 보안을 위한 응용에 PS/2 키보드를 사용할 것을 권유하고 있으나 보급률이 재빠르게 증가하고 있는 노트북 컴퓨터의 키보드는 PS/2 방식으로 연결되어 있음으로 볼 때 제기된 취약점은 매우 심각한 문제를 야기할 수 있어 이에 대한 신속한 대응이 필요하다.

본 논문은 상기의 PS/2 키보드 관련 하드웨어 취약점을 이용하는 키보드 감시 문제를 효과적으로 해결하기 위한 대응 방안을 제안하고 이를 구현하였다. 논문에서는 해당 하드웨어의 취약점을 해결하기 위한 다양한 시도들을 제시하고 각 시도들을 분석하며 이러한 시도들의 문제점과 보다 효과적인 해결책이 될 수 있는 방안에 대하여 자세히 서술한다.

본 논문의 구성은 다음과 같다. 제2장에서는 현재의 PC 플랫폼이 제공하는 키보드 관련 하드웨어의 구조를 설명하고 제3장에서 이러한 하드웨어가 가지는 취약점 및 시도 가능한 기존의 해결방법에 대하여 서술한다. 제4장에서는 해당 취약점을 효율적으로 해결하기 위하여 본 논문에서 제안하는 새로운 방안에 대하여 기술하며 제5장에서 제안하는 방안의 적용 시 고려하여야 하는 사항을 논하는 것으로써 결론을 맺는다.

Ⅱ. 키보드 하드웨어 구조

PC 플랫폼에서 사용자의 키보드 입력은 [그림 1]과 같은 구성을 통하여 이루어진다. 키보드는 사용자 키 입력에 대응하는 스캔코드라 불리는 코드를 만들어내는



〔그림 1〕 키보드컨트롤러와 키보드 연결

역할을 하는데 각 키는 서로 다른 스캔코드를 가지며 키를 누를 때와 뗄 때도 make code(1byte; 0xXY)와 break code(2bytes; 0xF0 0xXY)라는 서로 다른 스캔코드를 만들어 낸다. 이러한 스캔코드는 PC 플랫폼 내에서 키보드를 상대하는 전용하드웨어로서의 키보드컨트롤러에 전달되어 각각 1바이트씩의 make code (0x00~0x7F)와 break code (0x80~0xFF)로 생성되며 이들은 호스트프로세서에게 인터럽트 기구를 통하여 전달된다.

PC 플랫폼은 키보드컨트롤러로의 접근을 위하여 읽기와 쓰기가 가능한 컨트롤 포트와 데이터 포트의 두 포트를 준비하여 각각 0x64 번지와 0x60번지에 할당하고 있다. 이 두 I/O 포트의 기능을 요약하여 정리하면 [표 1]과 같다. 이들 포트를 통하여 키보드컨트롤러로의 제어코드나 제어인자, 키보드로의 명령코드, 명령인자 등을 전달하거나 키보드컨트롤러로부터의 제어응답, 키보드로부터의 명령응답 및 스캔코드 등을 수수한다.

제어코드는 키보드컨트롤러에 전달되고 해석되어 다양한 기능을 수행하며 추가적으로 인자를 요구하거나 응답을 제공하는 경우도 있다. 명령코드는 키보드컨트롤러를 통과하여 키보드로 전달되며 이 또한 인자나 응답을 요구하기도 한다. 스캔코드는 키보드로부터 전달되어 키보드컨트롤러를 통과하여 키보드 문자를 생성[3]하는 데에 활용된다.

키보드컨트롤러는 호스트프로세서와 상기의 다양한 정보를 주고받기 위하여 두 포트를 대상으로 값을 읽고 쓰기 위한 공유버퍼로서 입력버퍼와 출력버퍼를 가지고

〔표 1〕 키보드컨트롤러 포트의 역할

포트 이름 (주소)	읽기 (출력버퍼)	쓰기 (입력버퍼)
컨트롤 (0x64)	상태레지스터	제어코드
데이터 (0x60)	스캔코드 제어응답 명령응답	제어인자 명령코드 명령인자

PER	RTO	TTO	INH	C/D	SYS	IBF	OBF

비트	의미
PER	Parity Error
RTO	Receive Time Out
TTO	Transmit Time Out
INH	Inhibit
C/D	Control or Data
SYS	Warm or Cold
IBF	Input Buffer Full
OBF	Output Buffer Full

〔그림 2〕 키보드컨트롤러 상태레지스터

XLT	KTM	DAD	DKD	XLT	SYS	EAI	EKI

비트	의미
XLT	Unused
KTM	Keyboard Translate Mode
DAD	Disable Auxillary Device
DKD	Disable Keyboard
XLT	Unused
SYS	Warm or Cold
EAI	Enable Auxillary Interrupt
EKI	Enable Keyboard Interrupt

〔그림 3〕 키보드컨트롤러 설정레지스터

있다. 그러므로 소프트웨어는 입력버퍼를 통하여 키보드컨트롤러에게 정보를 써 넣고 출력버퍼를 통하여 키보드컨트롤러로부터 정보를 읽어낼 수 있다. 또한, 키보드컨트롤러는 키보드 제어 및 호스트프로세서와의 통신을 위하여 그 내부에 상태레지스터와 설정레지스터를 가지고 있다. 이들을 [그림 2] 및 [그림 3]에 보인다.

상기한 입력버퍼 및 출력버퍼를 통하여 정보전달을 하는 과정에서의 흐름제어를 위하여 키보드컨트롤러의 상태레지스터에는 OBF와 IBF 등의 플래그가 준비되어 있다. OBF와 IBF 비트는 각각 키보드의 출력버퍼 또는 입력버퍼에 데이터가 적재될 경우 1로 설정되며 해당 데이터를 읽어 낼 경우 0으로 지워진다. 단, 버퍼는 읽어 내더라도 기존에 적재된 정보가 그대로 남아 있어 이후에도 새로운 값이 적재되지 않는 한 재차 읽어낼 수 있다.

설정레지스터는 키보드의 동작 금지 또는 키보드컨트롤러로부터의 인터럽트 요청 금지 등의 주요 역할을 제어하기 위한 목적으로 사용되며 키보드컨트롤러의 제어코드를 이용하여 값을 읽어 내거나 새로운 값을 설정할 수 있다. 이를 위한 제어코드로는 0x20 및 0x60이 사용된다.

키보드컨트롤러를 위한 제어코드는 상기한 설정레지스터 조작 코드 이외에도 다양하게 존재한다. 이러한 제어코드는 키보드컨트롤러 및 키보드의 상태를 확인하는 등의 목적으로 이용하는데 제어코드가 제어인자 및 제어응답을 가지는지의 여부에 따라 크게 4가지로 분류될 수 있다. 즉, 인자나 응답이 없는 제어코드, 응답이 없이 인자만 가지는 제어코드, 인자 없이 응답만을 가지는 제어코드, 인자와 응답을 모두 가지는 제어코드가 있을 수 있다. [표 2]에 주요 제어코드와 이들 제어코드에 대한 인자 및 응답 유무 여부를 정리하였고 [그림 4]에 상위 소프트웨어가 이들 제어코드를 활용하여 구현하게 될 네 가지의 오퍼레이션의 예를 보였다.

Ⅲ. 키보드 하드웨어 취약점

키보드는 현재의 문자열 기반 패스워드 인증 방식에서는 핵심적인 도구이다. 그러나 기존 운영체제가 가지는 보안 구조의 한계로 인하여 직접 폴링에 의한 키보드 정보 유출이 가능하므로 이를 이용한 키보드 감시 소프트웨어들이 대거 등장하였다[1]. 이러한 감시 소프트웨어를 회피하기 위하여 그 동안 키보드컨트롤러 내

〔표 2〕 키보드컨트롤러의 주요 제어코드

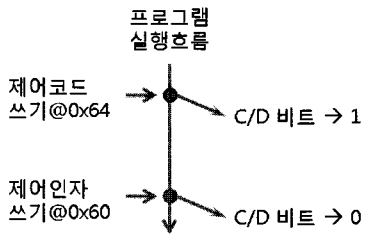
코드	종류	역할	인자	응답
20	Read Configuration Register	설정 레지스터를 읽어 출력버퍼에 적재	X	설정 레지스터 값
60	Write Configuration Register	인자 값을 설정 레지스터에 적재	O	X
AA	Self Test	키보드컨트롤러의 내부 진단	X	응답 값: 0x55
AB	Interface Test	Clock과 Data 선 검사	X	선 상태 값: 0x00
AD	Disable Keyboard Interface	Bit 4 Set (Clock 을 low로 유지)	X	X
AE	Enable Keyboard Interface	Bit 4 Clear (Clock 을 해방)	X	X
C0	Read Input Port	입력 포트를 읽어 출력버퍼에 적재	X	입력포트 값
D0	Read Output Port	출력 포트를 읽어 출력버퍼에 적재	X	출력포트 값
D1	Write Output Port	인자 값을 출력 포트에 출력	O	X
D2	Write Keyboard Output Buffer	인자 값을 출력버퍼에 적재	O	인자 값

```
1) Operation0: argument only
   while (IBF) ;
   Write (Some1) to Control Port ;  // C/D=1, IBF=1, OBF=X(0)
   while (IBF) ;
   Write (Some2) to Data Port ;     // C/D=0, IBF=1, OBF=X(0)

2) Operation1: reply only
   while (IBF) ;
   Write (Some1) to Control Port ;  // C/D=1, IBF=1, OBF=1
   while (!OBF) ;
   Read (Some2) from Data Port ;    // C/D=1, IBF=X(0), OBF=0

3) Operation2: both argument and reply
   while (IBF) ;
   Write (Some1) to Control Port ;  // C/D=1, IBF=1, OBF=X(0)
   while (IBF) ;
   Write (Some2) to Data Port ;     // C/D=0, IBF=1, OBF=1
   while (!OBF) ;
   Read (Some3) from Data Port ;    // C/D=0, IBF=X(0), OBF=0

4) Operation3: neither argument nor reply
   while (IBF) ;
   Write (Some1) to Control Port ;  // C/D=1, IBF=1, OBF=X(0)
```

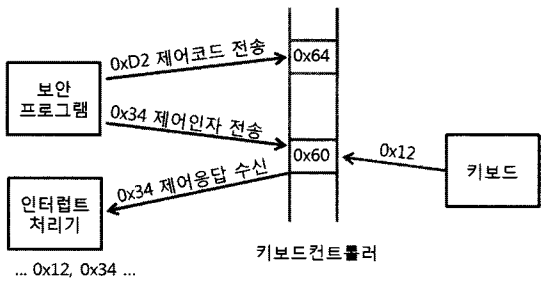
〔그림 4〕 키보드컨트롤러 제어코드 오퍼레이션의 분류

의 제어코드를 활용하는 방안이 시도되었다. 그 대표적인 것으로서 [그림 5]와 같이 0xD2 제어코드를 활용하여 무작위 코드 및 무작위 시간으로 스캔코드를 교란하는 방법이 있다[2,4]. 0xD2 제어코드는 인자를 전달하면 이 인자 값을 키보드로부터의 스캔코드와 마찬가지로 출력버퍼에 채워준다.

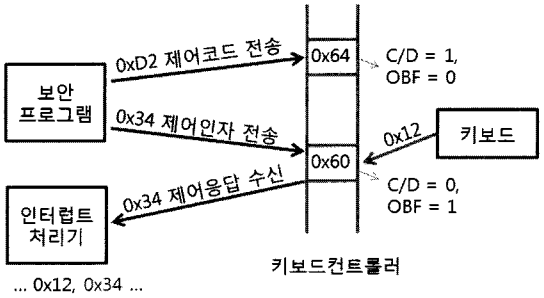
그러나 상기의 방법에 의하여 추가된 가짜 스캔코드는 키보드컨트롤러가 가지는 하드웨어 취약점으로 인하여 쉽게 구분 가능하다[2]. 키보드컨트롤러 상태레지스터 내의 C/D 플래그는 [그림 6]과 같이 입력버퍼를 통하여 전달되는 다양한 정보의 종류를 키보드컨트롤러가 구분할 수 있도록 호스트프로세서가 제공하는 최소한의 힌트로서 현재 입력된 정보가 어느 포트를 통하여 전달되었는지를 알려준다. 따라서 0xD2 제어코드를 이용하여 스캔코드를 교란하더라도 [그림 7]과 같이 제어코드



〔그림 6〕 제어코드와 C/D 비트와의 관계



〔그림 5〕 0xD2 제어코드에 의한 키보드 감시의 회피



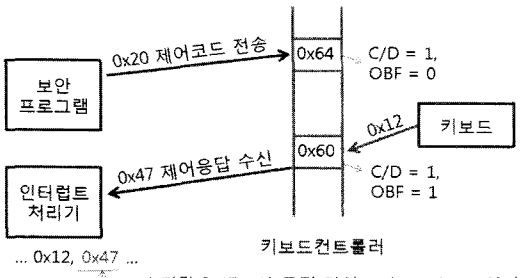
〔그림 7〕 감시 회피의 실패

0xD2를 이용하여 입력된 스캔코드는 C/D 비트의 폴링에지를 만들게 되므로 이를 검출하면 해당 스캔코드를 제거해 낼 수 있다.

그러나 만약 키보드컨트롤러에서 C/D 비트를 교란시킬만한 방법을 찾는다면 새로운 감시 방지 대책을 만들 수 있다. 즉, C/D 비트의 폴링에지가 발견되지 않으면서도 키보드컨트롤러의 출력버퍼로부터 입력되는 정보를 감시 프로그램이 스캔코드로 인식하게 하며 보안 프로그램은 해당 정보를 스캔코드와 구별할 수 있다면 이를 이용하여 스캔코드를 교란시킴으로써 감시를 방지할 수 있다. 이를 위하여 [그림 4]에서 인자를 가지지 않으면서 응답을 만들어 내는 Operation1 형태의 제어코드를 활용할 수 있다.

Operation1 형태의 특정 제어코드가 인자 없이 응답을 만들어 냄으로써 C/D 비트의 폴링에지 취약점이 해결되더라도 해당 응답이 예측 가능하여 스캔코드가 아님을 판별할 수 있다면 감시 프로그램은 이를 걸러낼 수 있다. 따라서 감시의 가능성을 최소화하는 문제는 예측 가능성이 가장 적은 응답을 가진 제어코드를 찾는 문제로 귀결된다.

Operation1 형태의 코드는 0xAA, 0xAB, 0xA9, 0x20, 0xC0, 0xD0 등이 존재한다. 이들 코드의 응답을 볼 때, 보안 프로그램은 스캔코드와 구별 가능하고 감시 프로그램은 스캔코드와 구별할 수 없는 값을 찾는 것은 쉽지 않다. 0x20 제어코드는 키보드컨트롤러 내의 설정



〔그림 8〕 추가적인 감시 회피 전략 및 결과

레지스터의 값을 읽어 내는 역할을 한다. 이 제어코드를 0xD2 제어코드와 함께 혼용하면 C/D 비트의 폴링에지를 교란시킴으로써 감시 프로그램을 속일 수 있을 것으로 보인다. 그러나 [그림 8]과 같이 이 코드에 의하여 읽혀지는 설정레지스터의 값은 제3번 비트인 SYS 비트를 제외하고는 장시간 사용자가 의도하는 값을 갖도록 할 경우 문제가 발생할 여지가 다분하며 SYS 비트조차도 이를 활용한 소프트웨어가 존재할 경우 변경하기 곤란하므로 다양한 값을 가질 수 없다. 따라서 감시 프로그램이 이를 무조건 걸러 내더라도 해당 스캔코드를 몇 개만을 잃게 되는 것으로 대개의 패스워드에 대하여 납득할 만한 수준으로 감시 가능하다. 더구나 설정레지스터의 최 상위 비트는 특수 비트로서 일반적으로는 0의 값을 가지므로 이 특성을 이용하면 스캔코드와의 구별이 쉬워진다. 즉, 최 상위 비트가 항상 0인 스캔코드가 수신된다는 것은 특정의 키 값을 위하여 항상 make code 만이 생성됨을 의미하므로 감시 프로그램에게 조금의 연산 오버헤드를 허락한다면 해당 코드의 추출 및 제거가 가능하여 효과적인 방안이 될 수 없다. 또한, 0xAA, 0xAB 등 기타의 제어코드도 그 응답이 더욱 단순하여 감시 방지를 위한 교란코드로 사용할 수 없다.

Ⅳ. 새로운 키보드 감시 방지 프로토콜

상기한 바와 같이 각 제어코드를 활용하여 감시 프로그램을 효과적으로 회피기는 쉽지 않다. 그러나 [그림 9]에서와 같이 복수의 제어코드를 연속하여 활용하면 C/D 비트의 폴링에지를 감추는 일이 가능해진다. 즉, 제3장에서 서술한 바와 같이 0xD2 제어코드를 이용하여 가짜 스캔코드를 생성한 직후에 Operation3 형태의 제어코드를 선택적으로 추가 수행하는 루틴을 구성하고 이를 임계영역으로 묶어 실행할 수 있다. 그 결과, Operation3 제어코드의 수행 여부에 따라 C/D 비트의 폴링에지가 결정되므로 C/D 비트를 교란시킬 수 있다.

```
ENTER_CRITICAL
    Operation2 with Some1 = 0xD2 and Some2 = Random ;
    // C/D = X --> 1 --> 0, Some3 = Random
    Operation3 with Some1 = (Selected) ; // Alternatively
    // C/D = 0 --> 1
EXIT_CRITICAL
```

〔그림 9〕 C/D 비트를 조작하여 감시 회피

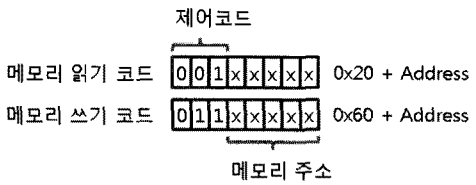
그러나 모든 Operation3 형태의 제어코드가 사용 가능한 것은 아니며 0xA6, 00xA7, 0xA8, 0xAD, 0xAE, 0xC1, 0xC2 등의 Operation3 제어코드 중에 시스템의 동작에 영향이 가장 적은 0xAE 제어코드가 효과적으로 활용 가능하다. 다만, 이러한 방법이 논리적으로 문제를 해결할 수는 있으나 프로그램의 구성으로 볼 때 임계영역 내에 네 번의 대기루틴을 가지게 되어 해당 프로세스가 운영체제가 제공하는 타임퀀텀[5]에 대하여 비교적 큰 시간을 배타적으로 실행하게 되므로 시스템에 따라 보안 프로그램이 과부하를 초래하는 경우가 간혹 발생할 수 있다. 그러므로 상대적으로 부하가 적은 해결 방안을 생각해 볼 수 있다.

키보드컨트롤러 내부에는 소정의 목적을 위하여 사용하도록 32바이트의 메모리를 준비하고 있다[6]. 이 메모리의 첫 번째 바이트가 키보드를 제어하기 위한 설정레지스터로 활용되며 그 나머지 31바이트는 현재의 PC에서는 사용되고 있지 않으므로 사용자가 임의의 목적에 활용할 수 있다.

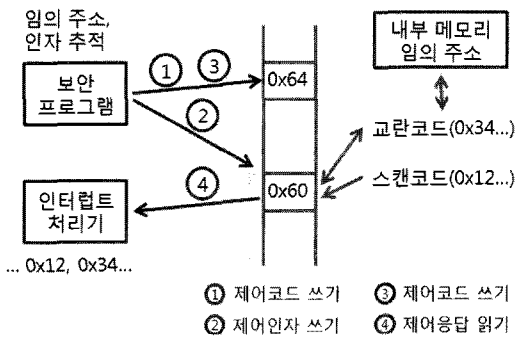
키보드컨트롤러의 설정레지스터는 제2장에서 서술한 바와 같이 제어코드 0x60을 사용하여 값을 쓰고 제어코드 0x20을 이용하여 저장된 값을 읽어낼 수 있다. 이 때, 제어코드의 하위 5비트를 변경하면 키보드컨트롤러 내부 메모리의 임의의 주소에 접근할 수 있다. 즉, [그림 10]과 같이 제어코드의 하위 5비트를 내부 메모리의 주소로 이용함으로써 메모리에의 저장을 위하여 0x60에서 0x7f, 메모리로부터의 판독을 위하여 0x20에서 0x3f의 제어코드를 만들어낼 수 있다.

위의 제어코드를 이용하여 키보드컨트롤러의 내부 메모리에 임의의 값을 쓰거나 읽는 것이 가능하다면 C/D 비트로부터 노출되는 취약점을 회피하면서도 보안 프로그램은 스캔코드와 구별할 수 있고 감시 프로그램은 스캔코드로 착각할 만한 응답을 생성하는 방안이 만들어질 수 있다. 즉, 보안 프로그램은 [그림 11]에서와 같이 자신만이 아는 내부 메모리 임의의 주소에 자신만이 아는 임의의 값을 쓰고 차후에 이를 읽어 냄으로써 감시 프로그램을 속일 수 있다. 여기에서 임의의 주소를 활용하는 이유는 혹시라도 정교히 작성된 감시 소프트웨어가 내부 메모리를 검사함으로써 보안 소프트웨어가 써 넣은 값을 검출하는 것을 피하기 위함이다. 물론, 임의의 주소에 임의의 값을 써 넣었다 할지라도 만일의 위험에 대비하여 별도의 처리기에 의하여 비주기적으로 메모리 전체의 값을 변경하는 것이 안전하다.

이를 위하여 작성된 의사코드의 일례를 [그림 12]와 [그림 13]에 보인다. [그림 12]의 Cheating routine에서와 같이 키보드컨트롤러의 내부 메모리 중 임의의 주소를 선택하여 그 곳에 0x60 제어코드를 이용하여 임의의 값을 저장하고 [그림 13]의 Juggling routine에서와 같이 Cheating routine에서 저장된 값을 0x20 제어코드를 이용하여 읽어 냄으로써 감시 소프트웨어를 교란시킨다. 이 때, 0x60 코드는 인자를 가지므로 C/D 비트의 폴링에지를 만들게 되나 응답이 없으므로 호스트프로세서가 읽어 갈 데이터는 발생하지 않으며 반대로 0x20 코드는 응답



〔그림 10〕 키보드컨트롤러 내부 메모리 읽고 쓰기



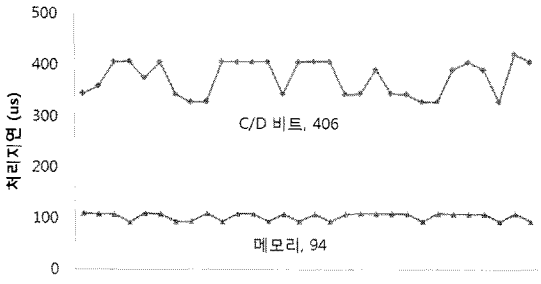
〔그림 11〕 제안 프로토콜의 동작

```
// Cheating routine ; sporadically repeated and recorded
ENTER_CRITICAL
   Operation0 with Some1 = (0x60+Random1) and Some2 = Random2 ;
   // C/D = 1 --> 0, Random2@Random1
EXIT_CRITICAL
```

〔그림 12〕 교란 코드를 준비하기 위한 루틴

```
// Juggling routine ; sporadically scheduled and repeated
ENTER_CRITICAL
   Operation1 with Some1 = (0x20+Random1) ;
   // C/D = 1 --> 1, Some2 = Random2
EXIT_CRITICAL
```

〔그림 13〕 교란 코드를 생성하기 위한 루틴



〔그림 14〕 C/D 비트 조작과 내부 메모리 활용 방안의 성능비교

이 있어서 호스트프로세서가 읽어갈 데이터가 발생하나 인자가 없으므로 C/D 비트의 변화가 발생하지 않는다.

또한 이 때 써 넣는 제어코드나 제어인자의 값은 휘발성 정보로서 써 넣은 후에 포트를 통하여 직접 추적할 수는 없으며, 다만 감시 프로그램이 키보드컨트롤러의 내부 메모리 전체를 검사하여 그 변화를 추적하면 논리적으로는 최후에 써 넣은 제어인자를 유추할 수 있다. 그러나 이는 처리지연이 매우 커서 기존 운영체제에서 그만큼 장시간의 배타적 접근권한을 보장 받지 못하므로 현실적으로 불가능하다. 따라서 제안한 방안을 이용하면 감시 프로그램을 효율적으로 회피할 수 있다.

본 논문에서 제안한 두 효과적인 키보드 감시 방지 프로토콜은 범용 8비트 프로세서를 이용하여 구현된 키보드컨트롤러를 상대로 프로그램 되므로 소정의 처리지연을 유발하게 된다[7]. 그러나 이러한 처리지연은 [그림 14]에서 보는 바와 같이 두 방안에 대하여 공히 심각한 수준은 아니며 특히, 내부 메모리를 활용한 방안은 100us 이하로 실행 가능하다.

V. 결론 및 향후과제

그 동안 기존 운영체제의 느슨한 접근권한 관리로 인하여 키보드로부터 입력되는 사용자 입력정보의 탈취를 방지하기가 매우 어려웠으며 이러한 문제를 해결하기 위한 접근방법들이 다수 시도되어 왔다. 그러나 이러한 시도는 키보드컨트롤러의 하드웨어 취약점으로 인하여 효율적으로 동작하기가 어려웠으며 사용자 정보의 유출 문제를 완벽하게 해결하지 못하였다. 본 논문에서는 이러한 문제를 해결하기 위하여 키보드컨트롤러의 하드웨어 취약점을 회피하는 방안과 키보드컨트롤러 내부 메모리를 활용하는 방안을 제안하였다. 본 논문에서 제안한 방안은 기존의 하드웨어 취약점이 존재하는 상황에서도 보안 프로그램이 감시 프로그램을 효율적으로 교란시킴으로써 감시 프로그램이 사용자 입력정보를 탈취하는 것을 효과적으로 방지할 수 있다. 이들을 활용하면 보안 프로그램은 감시 프로그램이 사용자 입력정보를 온전하게 탈취할 수 없도록 방해함으로써 사용자 입력정보의 유출을 방지할 수 있다.

참고문헌

[1] "키보드 해킹기법 및 대응기술 분석", 금융 ISAC, pp.8-10, 2005년 11월

[2] 배광진, 임강빈, "키보드 보안의 근본적인 취약점 분석", 한국정보보호학회 논문집 제18권 제3호, pp.89-95, 2008년 6월

[3] "Enhanced Super I/O Controller with Keyboard/Mouse Wake-up", Standard Microsystems Corporation, pp.119-130, Mar. 2000

[4] 태커스(주), "가상 데이터 전송을 이용한 키보드 해킹 방지 장치 및 방법", 대한민국특허청, 등록번호:10-0735727, 2007년 6월

[5] David A. Solomon, "Windows Internals", Microsoft Press, pp.340-344, Nov. 2006

[6] "PS/2 Model 50 and 60 Technical Reference", IBM Corporation, Chap.4, pp.7-18, Apr. 1987

[7] 임강빈, "키보드 보안", 유비쿼터스 정보보호 워크샵 2008, 2008년 5월

〈著者紹介〉

정 태 영 (Taeyoung Jeong) 학생회원
2007년 2월 : 순천향대학교 정보보호학과 학사
2007년 3월~현재 : 순천향대학교 정보보호학과 석사과정
<관심분야> 시스템보안, 운영체제보안, 임베디드시스템보안

임 강 빈 (Kangbin Yim) 종신회원
1992년 2월 : 아주대학교 전자공학과 학사
1994년 2월 : 아주대학교 전자공학과 석사
2001년 2월 : 아주대학교 전자공학과 박사
2003년 3월~현재 : 순천향대학교 정보보호학과 교수
2005년 3월~현재 : 한국정보보호학회 이사
<관심분야> 시스템보안, 운영체제보안, 임베디드시스템보안