Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.26, NO.6, Dec. 2016 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2016.26.6.1483

키보드를 모방하는 BadUSB를 차단하기 위한 보안 프로그램*

최 병 준,† 서 태 원‡
고려대학교

A Security Program To Protect against Keyboard-Emulating BadUSB*

Byung-jun Choi,† Taeweon Suh‡
Korea University

요 약

국내에 잘 알려지지 않은 Ducky USB는 키보드를 모방하는 BadUSB 중 하나로, Ducky USB를 활용한 대표적인 공격 방법은 PC에 삽입할 때마다 미리 설정한 키보드 입력이 자동으로 타이핑되도록 하는 것이다. 하지만 이를 차단하기 위한 이전 연구들은 USB의 성능을 몹시 저하시키거나 추가 장비를 필요로 하는 등의 문제점을 가지고 있다. 본 논문에서는 윈도우 운영체제 환경에서 특정 문자열의 입력 여부를 탐지하고 키보드 입력 간의 시간차를 활용하여 BadUSB의 실행을 차단할 수 있는 보안 프로그램을 개발하였고 실험을 통해 효과를 검증하였다.

ABSTRACT

Ducky USB is one of rarely-known keyboard-emulating BadUSBs. The attacking strategy using Ducky USB is taking and executing the scripted keystroke automatically whenever the USB is inserted into PC. Prior works exhibit some problems such as performance loss and additional device requirement. To solve this problem, this paper devised a countermeasure program to efficiently block the Duck USB in Windows. The experiment proves its effectiveness.

Keywords: BadUSB, Ducky USB, Countermeasure

I. 서 론

Universal Serial Bus (USB)는 컴퓨터 등의 정보기기에 주변 장치를 연결하기 위한 직렬 버스 규격의 하나로, 개인용 컴퓨터 주변기기에서 가장 많이 보급된 범용 인터페이스 규격이다[1]. 그럼에도 불구하고 국내에서 USB 보안은 다른 소프트웨어 혹은 하드웨어 보안에 비해 상대적으로 중요성이 강조되고 있지 않다. 해외의 경우 2014년 Blackhat 보안학회에서 Karsten Nohl 등은 8051 CPU 기반의 USB에 멀웨어(malware)를 탑재하고, 이 USB가 PC에 삽입되었을 때 어떠한 공격이 이루어질 수 있는지 시연하였다[2]. 이 발표에서는 악용의 여지 등을 이유로 펌웨어 소스코드를 공개하지는 않았다. 하지만 이후 Derbycon 4.0 보안학회에서 Adam Caudill 등은 이 문제를 해결하기 위해 노력하지 않는 USB 하드웨어 제조사에 압력을 가하기 위해 소스코드를 공개하였다[3]. 이처럼 악의적인 용도로 제작된 USB를 BadUSB라 칭하며 국내에 잘 알려지지 않은 Ducky USB는 키보드를 모방하는 BadUSB 중 하나이다. 이를 활용한 가장 대표적인 공격 방법은 PC에 USB를 삽입할 때마다 미리 설

Received(08. 08. 2016), Modified(10. 24. 2016), Accepted(11. 21. 2016)
* 본 연구는 2016년 정부(교육부)의 재원으로 한국연구재단의 기본연구지원사업 지원을 받아 수행되었습니다. (연구과제번호 NRF-2014R1A1A2059652)
† 주저자, cbj5210@nate.com
‡ 교신저자, suhtw@korea.ac.kr(Corresponding author)

정한 단축키와 문자열이 자동으로 타이핑되도록 하는 것이다. Ducky USB의 동작은 실제로 사람이 키보드 자판을 누르는 것인지, 악의적인 공격에 의한 키보드 입력인지 구분하기가 어렵기 때문에 현재로선 명확한 하드웨어/소프트웨어 솔루션이 제시되지 못하고 있다. 따라서 Ducky USB와 같은 키보드 모방 BadUSB를 악용하여 시스템에 치명적인 손상을 가하거나 개인 데이터를 탈취하려는 시도가 있을 수 있다. 본 논문은 키보드를 모방하는 모든 BadUSB의 실행을 차단하는 것을 목적으로 소프트웨어를 개발하였고, 그 효과를 실험을 통해 입증하였다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 상용 백신 프로그램을 이용하여 BadUSB를 차단할 수 있는지 여부와 관련 연구들을 소개한다. Ⅲ장에서는 기존에 알려진 USB의 취약성에 대해 서술하며 Ⅳ장에서는 BadUSB를 차단하기 위한 대응책(countermeasure) 프로그램을 제시한다. Ⅴ장에서는 Ⅳ장의 프로그램이 BadUSB를 차단할 수 있는지 실험하고 효과를 검증하였다. 마지막으로 Ⅵ장에서는 종합적인 연구 결과와 추후 연구에 대해 논의한다.

Ⅱ. 관련 연구

Fig. 1.은 USB의 내부구조로 크게 USB 컨트롤러와 플래시 두 부분으로 나눌 수 있다. 플래시 내부의 컨트롤러 펌웨어(controller firmware)는 추가적인 하드웨어 설치 없이 제조사나 판매회사 설계 요구사항에 의해 갱신 혹은 수정이 가능하다. 이런 가능성으로 인해 의도하지 않은 펌웨어의 위·변조가 가능하며, 이를 기반으로 BadUSB가 제작된다[5].

하지만 모든 버전의 Windows, Linux, Android OS는 자체적으로 BadUSB를 차단하는 기능이 없으며, USB 장치의 대용량 저장소(mass storage)만을 검사하는 상용 백신 프로그램들(AhnLab V3, 알약 등)과 USB 전용 보안 프로그램인 USB write protector, thumbscrew software USB write blocker, USBdeview, USB blocker, Trustport USB Antivirus, Safe USB 또한 BadUSB를 차단하지 못한다. Fig. 2.는 하나의 예를 보여주는 것으로, 오른쪽에 보이는 Windows Defender와 안티 바이러스 프로그램이 실시간 검사를 진행 중임에도 불구하고 키보드를 모방하는 BadUSB는 아무런 제약 없이 단축키를 입력하여 메모장을 실행하고 문자열을 타이핑하여 심슨 캐릭터 모양을 나타내고 있다[6].

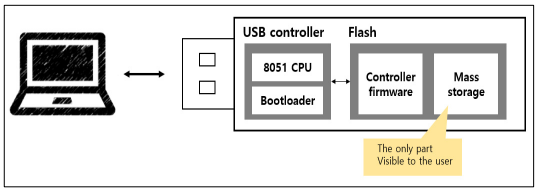

Fig. 1. 8051 CPU based USB Structure[4]

따라서 BadUSB를 차단하기 위한 여러 연구가 진행 중이며 그 중 첫 번째로는 독일의 보안 소프트웨어 회사인 G DATA가 개발한 USB Keyboard Guard[7]가 있다. 하지만 이 프로그램은 정상적인 키보드 장치를 PC에 연결해도 우선 차단한다는 단점이 있으며, Fig. 3.에 나타난 것처럼 위쪽에 표시되는 숫자를 마우스로 클릭하기만 하면 차단을 해제하거나 프로그램을 종료할 수 있다. 따라서 만약 악의적인 사용자가 물리적으로 시스템에 접근하는 경우 손쉽게 차단을 해제한 뒤 BadUSB를 사용할 수 있는 문제점이 있다.

두 번째 연구인 USBWall[6]에서는 BadUSB를 탐지하기 위해 Fig. 4.에서 보이는 것처럼 PC와 USB 장치 사이에 BeagleBone FPGA 보드를 연결하고 사용자의 승인을 얻은 USB 장치만 동작되도록 하였다. 하지만 USBWall은 BeagleBone과 같은 추가 장비가 필요하다는 것과 데이터가 FPGA를 거치기 때문에 심각한 전송 속도 저하를 유발한다는

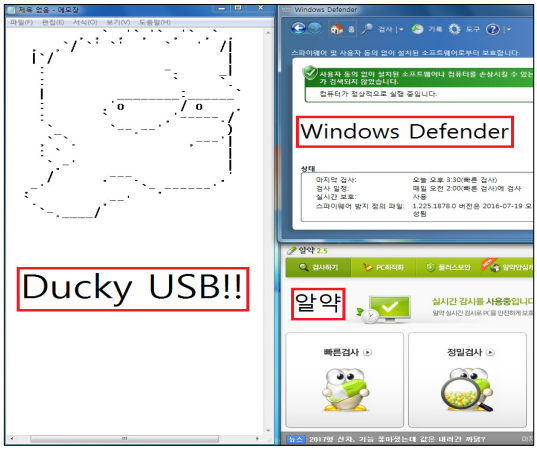

Fig. 2. BadUSB (Ducky USB) launched while Windows Defender and Alyac anti-virus program are in execution

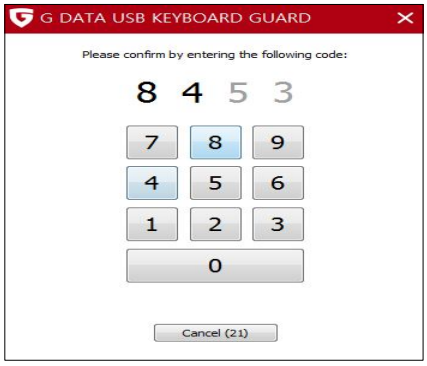

Fig. 3. G DATA USB Keyboard Guard

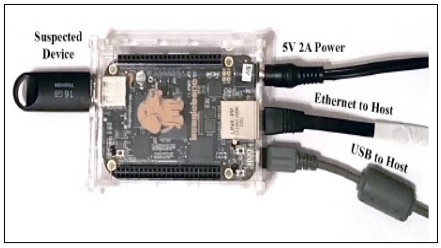

Fig. 4. USBWall structure

단점을 가지고 있다. 또한 만약 USBWall을 발전시켜 특정 하드웨어(예를 들어 Ducky USB)의 실행을 차단하는 방법을 개발하더라도 기존에 알려지지 않은 새로운 BadUSB가 등장할 경우 차단 효과를 기대하기 어렵다. 따라서 현재까지 효과적인 BadUSB 차단 솔루션이 존재하지 않는다고 볼 수 있다.

III. USB의 취약성

국내·외 다양한 언론 혹은 보안전문가들은 BadUSB 위협에 대해 지속적으로 논의해 왔고 대부분의 USB 컨트롤러 펌웨어 관련 제조사 및 보안업체들 또한 그 위험성을 인지하고 있지만 대응방안을 제시하지 못하고 있다. 더욱이 직접적인 피해 사례의 부재로 인해 향후 매우 심각한 보안 위협으로 대두될 BadUSB 기술이 일반인들에게는 인지되지 못하고 있다. 또한 BadUSB는 컨트롤러 펌웨어를 수정하여 제작되고 운영체제의 파일 시스템은 오직 대용량 저장소만을 보여주기 때문에 일반인은 BadUSB 기술을 인지하고 있더라도 본인 혹은 타인이 소유한 USB가 BadUSB인지 판단할 수 있는 방법이 없다. BadUSB를 사용함으로써 예상되는 피해는 다음과 같다[5].

- 키보드 자판 입력 조정 권한 획득
- 파일 관리자 비밀번호 등 중요정보 획득
- 파일 열람, 파일 변경 및 파일 추출
- 인터넷 접속 시 악성 웹사이트로 유도
- USB 포트에 접속되는 주변 장치들 감염
- 네트워크 카드 스푸핑, 컴퓨터 DNS 경로 조작
- PC 부트로더(boot loader) 감염
- USB 드라이브 내 hidden partition 생성

펌웨어 수정이 가능하여 BadUSB로 제작될 수 있는 8051 CPU 기반의 USB들[8]은 현재 판매가 중지 되었다. 하지만 BadUSB를 제작하는 기술은 8051 CPU 기반의 USB 컨트롤러 이외에도 적용이 가능한 기술이므로, USB 펌웨어 잠금 기능이 적용되지 않아 펌웨어를 임의로 수정할 수 있는 USB는 잠재적으로 BadUSB로 악용될 수 있는 가능성이 있다[5]. 본 논문의 실험에서 사용한 Ducky USB는 아래의 과정을 통해 키보드를 모방하는 BadUSB로 악용된다.

(1) 하나 또는 그 이상의 키 조합을 나타낼 수 있는 Ducky Script[9] 언어로 원하는 키보드 입력을 작성한다.
(2) Ducky Script 언어로 작성 된 텍스트 파일을 바이너리 파일로 인코딩하는 cross-platform Java duck encoder를 이용하여 이전 과정에서 작성한 Ducky Script를 Inject.bin 파일로 인코딩한다.
(3) Inject.bin 파일을 일반적인 Micro SD 카드에 옮긴다.
(4) Micro SD카드를 Ducky USB에 장착한 후 PC에 삽입하면 Inject.bin에 정의된 키보드 입력이 자동으로 타이핑된다.

해외 커뮤니티인 Hak5[10]에서 판매하는 Ducky USB의 외형은 Fig. 5.에서 보이는 것처럼 케이스를 장착하면 일반 USB와 차이가 없으며 상세한 사양은 Table 1.과 같다.

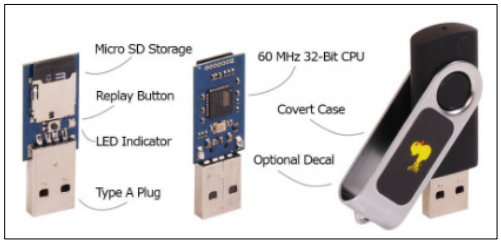

Fig. 5. Ducky USB[11]

Table 1. Ducky USB H/W specification[12]

Type	Specification
CPU	ATMEL AVR32 UC CPU
RAM	32KB
Flash memory	256KB
I/Os	Type A USB connector
	Onboard replay button

IV. 대응책 구현

본 논문에서는 키보드를 모방하는 BadUSB를 차단하기 위해 Ⅱ장에서 언급한 기존 연구들의 단점들을 보완하여 다음과 같은 기능들을 포함한 C++ 프로그램을 윈도우 API를 활용하여 개발하였다.

4.1 전역 후킹(Global Hooking)

윈도우 메시지(window message)[13]는 시스템과 윈도우 응용 프로그램이 상호 동작할 수 있도록 도와주는 매개체이다. 응용 프로그램이 생성한 실행창이 화면에 출력되고 사용자의 입력을 처리할 수 있는 것은 메시지가 존재하기 때문이다. 또한 윈도우의 각 스레드(thread)에는 스레드가 소유한 실행창을 제어하는 데 사용되는 메시지를 수신하기 위해 메시지 큐(message queue)가 부여된다. 따라서 사용자의 입력이 발생하면 메시지가 생성되며 생성된 메시지는 시스템 큐에 쌓이게 되고 이는 해당되는 스레드의 메시지 큐로 전달된다. 시스템은 상황에 따라서 수많은 메시지들을 발생시키는데 각 스레드는 자신이 수신한 메시지를 반드시 처리해야 할 의무가 있다. 따라서 스레드는 메시지 큐에서 전달받은 메시지를 꺼내서 메시지에 대한 동작이 정의되어 있는 콜백(callback) 함수를 호출하여 적절한 동작이 수행되도록 하거나 시스템에 메시지 처리를 요청한다[14].

특정 프로세스로 향하는 메시지를 가로채거나 수정하는 것을 메시지 훅(message hook)이라고 하며 실행중인 모든 프로세스와 앞으로 실행 될 모든 프로세스를 대상으로 메시지 훅을 수행하는 것을 전역 후킹이라고 한다. 본 논문의 프로그램은 전역 후킹 기능을 이용하여 현재 어느 프로세스가 활성화 되어 있는지와 관계없이 모든 키보드 입력 메시지(WM_KEYDOWN)를 감시할 수 있도록 하였다. 전역 후킹 기능을 수행하기 위해 사용한 함수는 Table 2.와 같다.

Table 2.의 함수에서 첫 번째 인자인 idHook은 추가할 훅 프로시저의 종류를 가리키며 키보드 전역 후킹을 위해서는 WH_KEYBOARD_LL(0x13)값을 가져야 한다. 두 번째 인자인 lpfn은 훅 프로시저의 포인터를 가리킨다. 즉 함수가 성공적으로 수행되면 훅 프로시저에 대한 핸들을 HOOKPROC의 형태로 lpfn에 반환된다. 세 번째 인자인 hMod는 훅 프로시저를 포함하는 DLL에 대한 핸들러로 훅 프로시저가 현재 프로세스 내에 위치한 경우는 null 값으로 설정되어야 한다. 마지막으로 dwThreadId는 훅 프로시저가 연결될 스레드의 식별자로 모든 스레드에 훅 프로시저를 적용하기 위해서는 값을 0으로 설정해줘야 한다. 따라서 실제 프로그램 소스코드에는 SetWindowsHookEx (WH_KEYBOARD_LL, HookCallback, NULL, 0)의 형태로 적용하였으며 전역 후킹을 포함한 프로그램의 모든 기능은 사용자의 일반적인 PC 사용에 방해가 되지 않도록

Table 2. SetWindowsHookEx function for global hooking[15]

```
HHOOK WINAPI SetWindowsHookEx(
  _In_ int       idHook,
     // The type of hook procedure to
        be installed.
  _In_ HOOKPROC  lpfn,
     // A pointer to the hook procedure.
  _In_ HINSTANCE hMod,
     // A handle to the DLL containing the
        hook procedure pointed to by the
        lpfn parameter.
  _In_ DWORD     dwThreadId
     // The identifier of the thread with
        which the hook procedure is to
        be associated.
);
```

백그라운드에서 동작하도록 하였다.

4.2 비정상적으로 빠른 문자열 입력 차단

파워쉘(powershell)[16]은 윈도우 7 이상의 버전에 기본으로 탑재된 쉘 프로그램이다. 파워쉘은 명령 프롬프트(cmd)와는 다르게 닷넷 프레임워크 스크립트를 실행시킬 수 있으며 로컬 시스템뿐만 아니라 원격 시스템을 제어하고 레지스트리 값을 수정할 수 있는 강력한 권한을 제공한다.

키보드를 모방하는 BadUSB는 키보드 입력을 통해 파워셸을 실행시킨 뒤, cat 명령어를 이용하여 배치파일을 생성하고 실행함으로써 멀웨어와 같은 동작을 수행할 수 있다. 본 연구에서는 키보드를 모방하는 모든 BadUSB가 스크립트(script)와 비슷하게 동작하기 때문에 문자열이 빠르게 입력될 것이라는 가정을 세웠다. 이를 검증하기 위해 clock() 함수를 이용하여 각각의 키보드 입력 사이에 걸리는 시간을 측정하였으며 측정된 값을 통해 사람과 키보드를 모방하는 BadUSB를 구분할 수 있는 기준점을 찾고자 하였다. 일반적으로 사람이 문자열을 가장 빠르게 타이핑하는 속도는 약 1200cpm(characters per minute)으로 한 번의 키보드 입력 당 평균적으로 0.05초가 소요된다고 알려져 있다[17]. 하지만 실험 결과 2개 이상의 자판을 거의 동시에 누를 때(예를 들어 단축키를 입력할 때) 키보드 입력 간 시간차는 0.001초보다 조금 긴 시간들이 측정되었다. 따라서 각각의 키보드 입력 사이에 걸리는 시간이

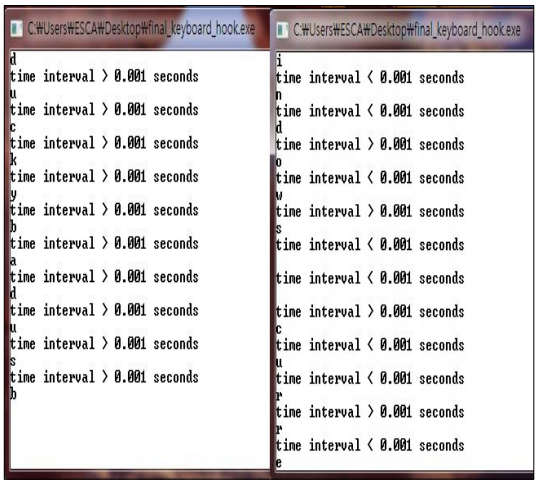

Fig. 6. Time interval measurement: (left) human key input (right) Ducky Script key input

Table 3. LockWorkStation function[18]

```
BOOL WINAPI LockWorkStation(void);
    // The LockWorkStation function is
       callable only by processes running
       on the interactive desktop.
       In addition, the user must be logged
       on, and the workstation cannot
       already be locked.
```

0.001초보다 오래 걸리는지 아닌지를 판단하는 프로그램을 만들었으며 이 프로그램으로 사람이 키보드를 누를 때와 Hak5에서 제공하는 Ducky Script 예제를 수행할 때 키보드 입력 사이에 걸리는 시간차를 조사해보면 Fig. 6.과 같은 결과를 얻을 수 있었다. Fig. 6.의 왼쪽은 사람이 연속적으로 'duckybadusb'를 타이핑하였을 때 각각의 키보드 입력 사이의 시간이 0.001초 이상인지를, 오른쪽은 Ducky USB가 Ducky Script 예제를 수행하였을 때 키보드 입력 간의 시간이 0.001초 이상인지를 나타낸 것으로 사람이 입력했을 때와는 다르게 0.001초 미만의 키보드 입력이(부등호 방향이 다른 것이) 상당 수 많음을 알 수 있다. 이에 착안해서 본 논문의 프로그램은 전역 후킹을 통해 비정상적으로 빠른(키보드 입력 간 시간차가 0.001초 미만인) 키보드 입력 메시지가 지속적으로 확인되면 Table 3.의 API를 호출하도록 하였다.

Table 3.의 LockWorkStation 함수는 호출되는 즉시 윈도우 운영체제를 잠금 상태로 변경하며 매개변수를 가지지 않는다. 잠금을 해제하기 위해서는 로컬 윈도우 계정 아이디와 패스워드를 이용하여 윈도우에 다시 로그인해야하기 때문에 계정 정보를 모르는 악의적인 사용자는 이를 해제할 수 없다. 또한 잠금을 해제하기 전의 모든 키보드 입력은 시스템에 영향을 미치지 못하기 때문에, 시스템의 상태 변화를 고려하지 못한 채 연속으로 키보드 입력을 하는 BadUSB를 자연스럽게 차단할 수 있다.

4.3 특정 문자열 입력 차단

만약 악의적인 사용자에게 시스템을 공격할 시간이 충분히 주어진다면 Ducky Script의 딜레이 기능을 활용함으로써 강제로 키보드 입력 사이에 시간차를 추가하여 4.2 방식의 차단을 피해갈 수 있다. 따라서 본 논문의 프로그램은 일반적으로 많이 사용

되지는 않지만 시스템 공격에 활용될 수 있는 특정 문자열(예를 들어 'powershell')이 Fig. 7.의 유한 상태 오토마타처럼 연속적으로 입력되는 것이 탐지되면 LockWorkStation 함수를 이용하여 윈도우 운영체제를 잠금 상태로 변경하도록 하였다.

하지만 Fig. 7.의 탐지 방법만을 사용하면 어떠한 문자열을 순서대로 입력하지 않고 방향키와 home, end키 등을 활용하여 입력순서를 변형하는 경우를 탐지하기 어렵다. 예를 들어 owershll을 먼저 입력한 후 home키를 누른 뒤에 p를 입력하면 동일하게 powershell을 입력할 수 있지만 Fig. 7.의 유한상태 오토마타 탐지를 우회할 수 있다. 따라서 본 논문의 프로그램은 문자열 입력 우회를 탐지할 수 있는 Algorithm1을 포함하고 있다. 키보드를 모방하는 BadUSB는 스크립트 명령어를 입력함으로써 동작하므로 Algorithm1은 엔터키가 입력되는 것을 기준으로 키보드 입력을 나눠 특정 문자열이 입력되었는지를 판별한다. Algorithm1에서 사용되는 변수들의 의미는 Fig. 8.에 나타나 있다.

position은 현재 입력되고 있는 커맨드 라인에서 커서의 위치를 나타내며 초기 상태인 가장 왼쪽에서의 값은 -1이고 오른쪽으로 한 칸 움직일 때마다 1씩 증가한다. string_length는 현재 커맨드 라인에 작성된 문자열의 길이를 나타내며 문자열이 입력되지 않은 초기의 값은 -1이다. current_string은 현재 커맨드 라인에 입력된 문자열을 저장하며 current_string에 BadUSB 판별에 사용되는 문자열이 포함되어 있다면 LockWorkStation 함수를 호출하도록 하였다. 또한 previous_string과 previous_string_length는 가장 최근에 입력된 엔터키로 인해 수행된 명령어와 명령어의 길이를 나타낸다. 이 2개의 변수들은 다음과 같은 우회 문자열 입력을 탐지하기 위해 사용된다. 커맨드 라인에서 "↑"키가 입력되면 현재 커맨드 라인에 입력된 문자열을 지우고 가장 최근에 수행된 명령어를 커맨드 라인에 다시 나타낸다. 이를 통해 po를 타이핑하고 엔터키를 누른 뒤에 "↑"키를 입력하고 wershell을 타이핑하면 "powershell"을 커맨드 라인에 나타낼 수 있다. 따라서 본 논문의 String detection 알고리즘에서는 "↑"키가 입력되면 previous_string과 previous_string_length를 활용하여 새로 입력되는 문자열과의 조합하는 과정을 통해 위와 같은 우회 입력도 탐지할 수 있도록 하였다. 그 외의 다른 특수키에 대한 처리 역시 Algorithm1에 나타나 있다.

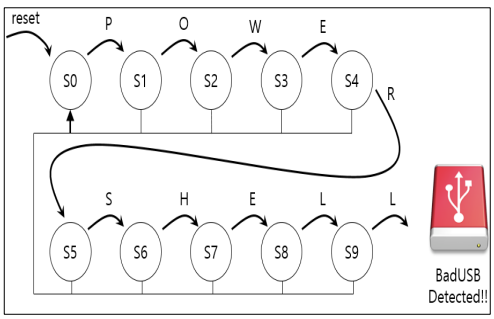

Fig. 7. Finite State Automata to check whether particular character string (e.g. powershell) is entered

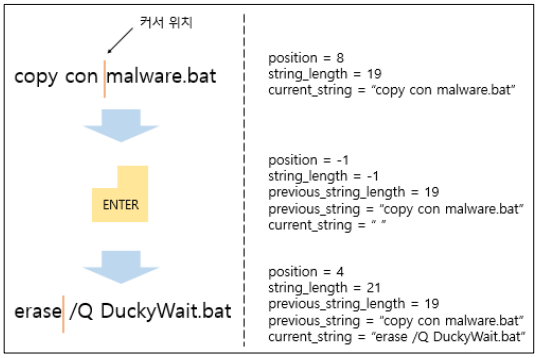

Fig. 8. String detection algorithm parameters

Algorithm1 *string detection*

```
1: position ←-1, string_length ←-1,
   previous_string_length ←-1
2: key ←" ", current_string ←" ",
   previous_string ←" "
3: Repeat 4~32 for every keyboard input event
4:    if key==character
5:    then position++, string_length++
      current_string.insert(position, string_length)
6:    else
7:       if key==home && position!=-1
8:          then position ←-1
9:       else if key==end && position!=-1
10:         position ←string_length
11:      else if key==back_space && position!=-1
12:         current_string.erase(position, 1)
13:         position--
14:      else if key==del && position!=string_length
```

```
15:        current_string.erase(position+1, 1)
16:        string_length −
17:     else if key=="←" && position!=-1
18:        position −
19:     else if key=="→" && position!=string_length
20:        position ++
21:     else if key=="↑"
22:        position ←previous_string_length
23:        string_length ←previous_string_length
24:        current_string ←previous_string
25:     else if key==enter
26:        previous_string_length ←string_length
27:        position ←-1
28:        string_length ←-1
29:        previous_string ←current_string
30:        current_string ←" "
31:  if (current_string.find("powershell"))
32:     then LockWorkStation();
```

4.4 프로그램 종료 방지

이전 연구인 G DATA Keyboard Guard와는 다르게 본 연구에서는 악의적인 사용자가 BadUSB 차단 프로그램을 종료하고 재차 BadUSB를 삽입하여 시스템을 공격하는 것을 방지하기 위해 사용자가 프로그램 종료를 요청하면 Table 4.의 함수를 호출하도록 하였다. LogonUser 함수의 주요 매개변수는 2 바이트 유니코드 문자열 변수(LTPSTR)인 lpszUsername과 lpszPassword이며 이들은 각각 현재 로그인 되어 있는 로컬 윈도우 계정의 아이디와 패스워드에 대응된다. 또한 lpszUsername과 lpszPassword에 올바른 윈도우 계정정보가 입력되지 않는다면 LogonUser 함수는 null값을 반환한다. 따라서 본 논문의 프로그램은 LogonUser 함수의 반환값이 null인 경우 현재 시스템을 소유한 사람이 정당한 사용자가 아니라고 판별하여 Fig. 9.처럼 프로그램이 종료되지 않도록 하였다.

Table 4. LogonUser function[19]

```
BOOL LogonUser(
  _In_      LPTSTR  lpszUsername,
    // A pointer to a null-terminated string
       that specifies the name of the user.
  _In_opt_ LPTSTR  lpszDomain,
  _In_opt_ LPTSTR  lpszPassword,
    // A pointer to a null-terminated string
       that specifies the plaintext password
       for the user account specified by
       lpszUsername.
  _In_      DWORD   dwLogonType,
  _In_      DWORD   dwLogonProvider,
  _Out_     PHANDLE phToken
);
```

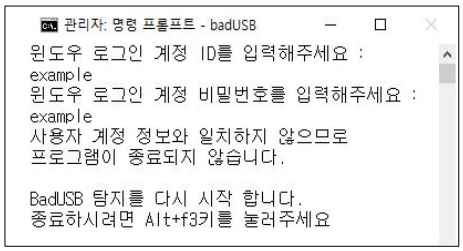

Fig. 9. Windows user authentication preventing the countermeasure shutdown

V. 실험 및 성능 분석

5.1 BadUSB 차단 여부

본 논문의 프로그램이 정상적으로 BadUSB를 차단할 수 있는지 확인하기 위해 Ducky USB에 Hak5에서 제공하는 Ducky Script 예제[20]와 직접 작성한 'Hide_attack_reg' Ducky Script를 탑재하여 테스트를 진행하였다. Fig. 10.은 Hide_attack_reg의 대략적인 동작 방법을 보여준다. Inject.bin에 의해 PC의 %temp% 폴더에 invis.vbs 파일과 악의적인 배치파일들을 생성하고 실행한다. invis.vbs는 배치파일들이 은밀하게 수행될 수 있도록 powershell 창을 투명하게 만들어 모니터 화면에 나타나지 않도록 하는 비주얼베이직 스크립트 파일이다. 배치파일들은 드라이브 이름이 'DUCKY'인 USB를 삽입하면 PC의 '내 문서' 폴더 내 모든 파일들이 해당 USB로 복사되도록 하며 윈도우 레지스트리를 수정하여 Ducky USB를 제거하고 PC를 재부팅하더라도 다시 배치파일들이 실행되도록 하는 기능을 포함하고 있다. 실험 결과 Table 5.에서 보여주는 것처럼 본 논문의 프로그램이 키보드를 모방하는 BadUSB를 100% 차단할 수 있음을 확인할 수 있다. 사용한 Ducky Script들과 본 논문의 프로그램 소스 코드는 https://github.com/cbj5210/BadUSB/에 업로드 되어있다.

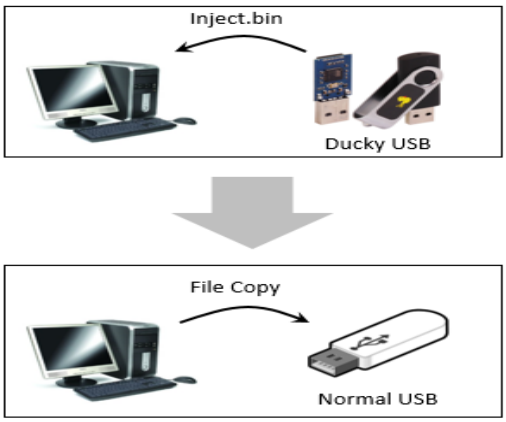

Fig. 10. Hide_attack_reg procedure

Table 5. Ducky USB blocking outcome by devised countermeasure

Ducky Script Payload	blocking BadUSB?
Hello World	success
WiFi password grabber	success
Hide CMD Window	success
Reverse shell	success
Fork bomb	success
Powershell wget execute	success
Website Lock	success
Windows 10 : Download & Change Wallpaper	success
Hide_attack_reg	success

5.2 벤치마크를 이용한 오버헤드 측정

본 논문의 프로그램이 USB의 정상적인 접근 속도에 미치는 영향을 파악하기 위해 ATTO Disk[21], CrystalDiskMark[22] 벤치마크 프로그램을 Table 6.의 환경에서 수행하였다. ATTO Disk와 CrystalDiskMark는 USB의 데이터 전송 단위인 블록 크기(4KB, 8KB 등)에 따른 USB의 읽기, 쓰기 접근 속도를 MB/s 단위로 측정하며 Fig. 11.은 벤치마크 프로그램 수행결과를 보여준다. 본 논문의 프로그램이 실행중일 때(with countermeasure)와 아닐 때(without countermeasure) 벤치마크 측정 결과가 거의 차이가 없음을 알 수 있다. 따라서 본 논문에서 개발한

Table 6. Experiment environment

Device	Type	Specification
Host PC	CPU	Intel Core 2 duo E8400 3.0GHz
	RAM	2GB
	OS	Windows 7 Enterprise K 64bit
	USB version	USB 2.0
USB Device	Normal USB	TG Pastel 32GB

프로그램은 기존 연구인 USBWall와 다르게 USB 접근 속도에 악영향을 미치지 않는다는 것을 확인할 수 있다. 또한 본 논문의 프로그램은 78KB의 저장 공간과 708KB의 메모리만을 필요로 하며 모든 시스템 메시지를 후킹하는 것이 아닌 키보드 메시지만을 선택적으로 후킹한다. 따라서 백그라운드에서 계속 동작하더라도 시스템에 부담을 주지 않는다는 장점이 있다. 이를 검증하기 위해 본 논문의 프로그램이 실행중일 때와 아닐 때 시스템 성능을 측정하는 벤치마크 프로그램을 수행하였다. 사용한 벤치마크 프로그램은 GeekBench4[23]로, GeekBench4

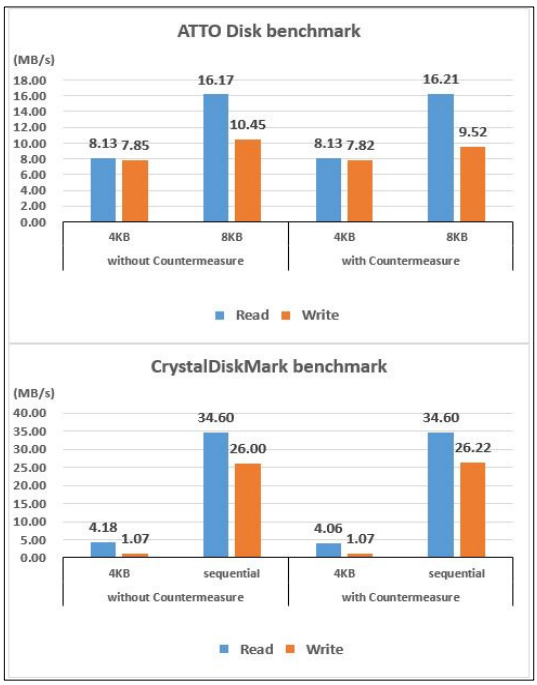

Fig. 11. Benchmark performance with and without countermeasure

Score는 성능에 비례하므로 Score가 2배 크다면 성능도 2배 좋다고 기대할 수 있다. Fig. 12.는 GeekBench4 벤치마크 프로그램 수행결과를 보여준다. 벤치마크 측정 결과의 차이가 거의 없으므로 본 논문에서 개발한 프로그램이 USB 접근 속도뿐만 아니라 시스템 성능에도 유의미한 악영향을 미치지 않는다는 것을 확인할 수 있다.

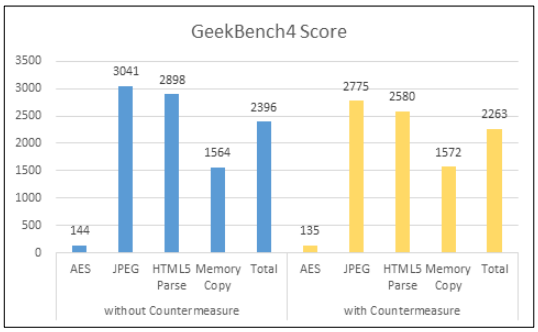

Fig. 12. GeekBench4 Score with and without countermeasure in Windows OS

5.3 논의

본 논문에서 개발한 프로그램은 Table 5.에 보인 것과 같이 키보드를 모방하는 BadUSB를 차단하는 실험에 대해 유효성을 입증하였으나 다음과 같은 특이 사항에 대해 생각해 볼 수 있다. 본 논문의 프로그램은 키보드 입력 간의 시간차를 계산하여 차단 기능을 구현하였으나 악의적인 사용자에게 시스템을 공격할 수 있는 많은 시간이 주어진다면 키보드 입력 사이에 딜레이를 둠으로써 탐지를 피해갈 수 있다. 하지만 이는 사람이 직접 키보드를 입력하는 것보다 신속하게 악의적인 키보드 입력을 수행하려는 BadUSB의 제작 의도와 모순되므로 BadUSB 차단을 위해 고려할 사항이 아니다. 또한 특정 문자열 입력 탐지 기능을 통해 프로그램의 기능을 보완할 수 있도록 하였다.

VI. 결론 및 향후 연구

키보드를 모방하는 BadUSB는 낮은 인지도, 구체적인 피해 사례의 부재로 인해 대응방안이 시급하게 논의되진 않았다. 하지만 아직까지 피해 사례가 등장하지 않았을 뿐, 추후 언제든 BadUSB를 악용하여 시스템에 치명적인 손상을 가하거나 개인적인 데이터를 탈취하려는 시도가 있을 수 있다. 본 논문에서는 윈도우 운영체제 환경에서 키보드를 모방하는 BadUSB의 실행을 차단하는 C++ 프로그램을 윈도우 API를 활용하여 개발하였다. 또한 본 논문의 프로그램을 실행하고 다양한 Ducky Script를 Ducky USB에 탑재하여 PC에 삽입하는 실험을 통해 차단 효과를 검증하였다. 실험 결과 본 논문의 프로그램은 Ducky USB의 실행을 100% 차단하였으며 오버헤드가 거의 없음을 보여주었다.

향후에는 평소 사용자의 키보드 입력 간 시간차 패턴을 파악하여 BadUSB 탐지에 활용하는 기능을 추가함으로써 본 논문의 프로그램을 개선할 것이다. 또한 Linux, Android 운영체제 환경에서 키보드를 모방하는 BadUSB를 차단하는 프로그램을 개발할 것이다.

References

[1] http://www.doopedia.co.kr/doopedia/master/master.do?_method=view&MAS_IDX=150923001512817
[2] https://www.blackhat.com/us-14/briefings.html#badusb-on-accessories-that-turn-evil
[3] https://adamcaudill.com/2014/10/02/making-badusb-work-for-you-derbycon/
[4] https://srlabs.de/blog/wp-content/uploads/2014/07/SRLabs-BadUSB-BlackHat-v1.pdf
[5] Jun Choi, "Countermeasures for badusb vulnerability," Journal of the Korea Institute of Information Security and Cryptology, 25(3), pp. 559-565, June. 2015
[6] Kang Myung, "USBWall: A Novel Security Mechanism to Protect Against Maliciously Reprogrammed USB Devices," M.S. Thesis, University of Kansas, December 2015.
[7] https://www.gdatasoftware.com/en-usb-keyboard-guard
[8] https://github.com/brandonlw/Psychson/wiki/Known-Supported-Devices
[9] https://github.com/hak5darren/USB-R

ubber-Ducky/wiki/Duckyscript
〔10〕 https://www.hak5.org/
〔11〕 http://hakshop.myshopify.com/products/usb-rubber-ducky-deluxe?variant=353378649
〔12〕 https://github.com/hak5darren/USB-Rubber-Ducky/wiki/Hardware
〔13〕 https://msdn.microsoft.com/ko-kr/library/windows/desktop/ff381405.aspx
〔14〕 Sang-min Jung, Sung-il Nam, and Tae-hoon Kim, Application Hacking, Book and Life, pp.18-27, June. 2009
〔15〕 https://msdn.microsoft.com/en-us/library/ms644990(VS.85).aspx
〔16〕 https://msdn.microsoft.com/en-us/powershell
〔17〕 https://www.youtube.com/watch?v=frSIJpa3w1I
〔18〕 https://msdn.microsoft.com/en-us/library/aa376875(VS.85).aspx
〔19〕 https://msdn.microsoft.com/en-us/library/aa378184(VS.85).aspx
〔20〕 https://github.com/hak5darren/USB-Rubber-Ducky/wiki/Payloads
〔21〕 https://www.attotech.com/disk-benchmark/
〔22〕 http://crystalmark.info/
〔23〕 http://geekbench.com/

〈저자소개〉

최 병 준 (Byung-jun Choi) 학생회원
2015년 2월: 고려대학교 컴퓨터교육과 학사
2015년 3월~현재: 고려대학교 컴퓨터학과 석사과정
〈관심분야〉 컴퓨터구조, 임베디드 시스템, 컴퓨터보안

서 태 원 (Taeweon Suh) 정회원
1993년 2월: 고려대학교 전기공학과 학사
1995년 2월: 서울대학교 전자공학과 석사
2006년 12월: Georgia Institute of Technology, Computer Engineering 박사
2007년 2월~2008년 8월 Intel Corporation, System Engineer
2008년 9월~현재: 고려대학교 컴퓨터학과 교수
〈관심분야〉 컴퓨터구조, 임베디드 시스템, HPC, 컴퓨터보안