안전하고 신뢰성있는 전자상거래를 위한 키보드 입력 보안 시스템의 설계 및 구현

최 성 욱[†] · 김 기 태[††]

요 약

최근 인터넷을 통한 전자상거래의 이용이 날로 급증하고 있지만, 만약 이를 이용하는 사용자의 PC 내에 키보드 입력을 모니터링하는 해킹 툴이 설치되어 있다면 사용자의 입력 내용을 가로챌 수 있게 되어 개인정보가 유출될 위험성을 갖게 된다. 본 논문에서는 이러한 키보드 입력 정보가 유출되지 않도록 보안하는 보안 시스템을 설계하고 구현하였다. 본 논문의 핵심은, 키보드 입력 시 발생하는 키보드 인터럽트 신호를 가로채어 키보드 입력을 암호화한 후, 필요에 따라 브라우저에 임베딩되어 있는 응용프로그램에서 이를 복호화하여 화면에 복호화한 값을 보여주고, 암호화된 값을 웹서버로 전송하여 웹서버단에서 복호화를 수행하여 처리하자는 것이다. 또한 브라우저에 대한 직접적인 해킹 공격을 막기 위해 자체 제작된 입력 컨트롤을 개발하였으며, 현재 출시되어 있는 키보드 보안 제품들과 달리 사용자의 입력 데이터가 최종적으로 웹서버에 도달하기 전까지 최소한 *와 같은 별표(asterisk)문자로 표시되는 패스워드 필드는 복호화하지 않도록 개선되었으며, 이로 인해 보다 안전한 고객 정보 보호 시스템을 실현할 수 있는 가능성을 제시하였다.

키워드 : 전자상거래, 보안, 키보드, 입력, 암호화

Design and Implementation of a Keyboard Input Security System for Safe and Trusted E-Commerce

Sung-Wook Choi[†] · Ki-Tae Kim[††]

ABSTRACT

It is growing to use the E-Commerce, recently. However, if a cracking tool that detects the keyboard input is set up, users' input values and personal information could be taken away. This paper shows the design and implementation of security system that prevent the keyboard input information leaking. The ideas of this paper are encrypting the keyboard input values with using the keyboard interrupt hooking, the browser embedding program's decrypting the values in case of need and decrypting all values in the web server. The own input control was developed for direct attacks to the browser, and that the values of password fields which are showed as *(asterisk character) won't be decrypted in the client PC is different from other commercial keyboard input security systems. Consequently, this paper shows the chance of realizing a lot safer customer information protective system than before.

Key Words : E-Commerce, Security, Keyboard, Input, Encryption

1. 서 론

인터넷이 보편화된 현대사회에서 전자상거래는 우리 사회에 없어서는 안 될 중요한 요소로 자리 잡고 있다. 이러한 전자상거래의 예로는 인터넷 뱅킹과 같은 금융 서비스나 인터넷 쇼핑몰, 각종 커뮤니티 사이트 등을 들 수 있으며, 이용자는 키보드나 마우스를 통해 자신의 의견이나 정보를 표출함으로써 이러한 서비스를 이용하게 된다. 이 때, 대부분의 전자상거래 상에서 이용자는 본인의 아이디나 비밀번호 등 자신의 개인 정보를 키보드를 통해 입력하게 되는데, 때에 따라서는 은행 계좌번호나 계좌비밀번호, 주민번호, 신용카드번호 등과 같이 개인의 자산이나 프라이버시와 관련된 중요한 개인 정보를 입력하는 경우가 종종 생긴다.

현재 대부분의 전자상거래 서비스에는 공개키기반구조(PKI)나 보안소켓레이어(SSL)를 통한 네트워크 구간의 암호화 솔루션이 적용되어 있기 때문에 통신구간에서의 해킹시도를 무력화할 수 있기 때문에 인터넷 상에서 전송되는 고객 정보에 대한 보안은 유지할 수 있다. 그러나 로컬PC 내에 키보드 입력을 모니터링하는 해킹툴이 설치되어 있다면 네트워크 구간으로 전송 이전에 사용자의 입력 내용을 가로챌 수 있게 되어 개인정보가 유출될 위험성을 갖게 된다[1].

† 준 회 원 : 중앙대학교 컴퓨터공학과 박사과정
†† 정 회 원 : 중앙대학교 공과대학 컴퓨터공학부 명예교수
논문접수 : 2005년 8월 18일, 심사완료 : 2005년 12월 9일

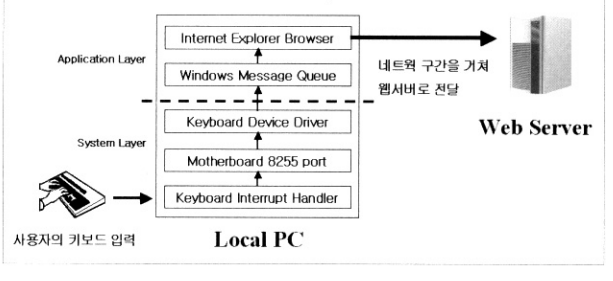
본 논문에서는 이러한 고객이 입력한 개인 정보가 안전하게 서버로까지 전달될 수 있도록 하는 시스템을 개발함으로써 기존이 알려진 각 해킹구간에서 입력 정보가 안전함을 보여준다. 또한 기존의 입력 정보 보안시스템과 비교하여 본 시스템이 갖는 장점을 제시한다.

제2장에서는 기반연구로서 키보드 입력 정보가 흘러가는 과정과 본 논문에서의 시스템의 기반 기술과 관련된 키보드 인터럽트에 대한 이론과 AES Rijndael 암호화 알고리즘에 대해 소개한다. 제3장에서는 본 논문에서 제안하는 키보드 입력 보안 시스템의 설계 및 구현을, 제4장에서는 구현 결과 분석과 기존 시스템과의 비교분석을 해보고 제5장에서는 결론과 향후 연구과제를 제시한다.

2. 관련 연구

2.1 키보드 입력 정보의 흐름

윈도우 운영체제와 인텔(Intel)의 x86계열 프로세서(CPU)가 탑재되어 있는 PC상에서 전자상거래를 이용하는 이용자가 키보드를 통해 자신의 개인 정보를 입력하고 그 결과가 화면에 전송되어 네트웍을 통해 웹서버로 전달되는 과정은 일반적으로 (그림 1)과 같다.



(그림 1) 키보드 입력 데이터의 흐름

즉, 키보드 입력을 하는 순간 키보드 인터럽트가 발생하여 이를 처리하는 인터럽트 핸들러를 거쳐 시스템 레이어의 키보드 드라이버를 통해 응용 프로그램 레이어에 전달되어 윈도우 메시지 형태로 시스템 큐에 저장되어 있으며, 입력을 한 해당 브라우저에게 전달되어 화면에 출력된다.[2] 또한 입력 후 사용자가 제출(submit) 버튼과 같은 확인 버튼을 하는 클릭하는 순간 HTML의 <form>태그를 통해 GET 혹은 POST 방식으로 입력 데이터가 네트웍을 통해 HTTP 프로토콜 형태로 클라이언트 PC로부터 웹서버까지 전달된다.

2.2 키보드 인터럽트

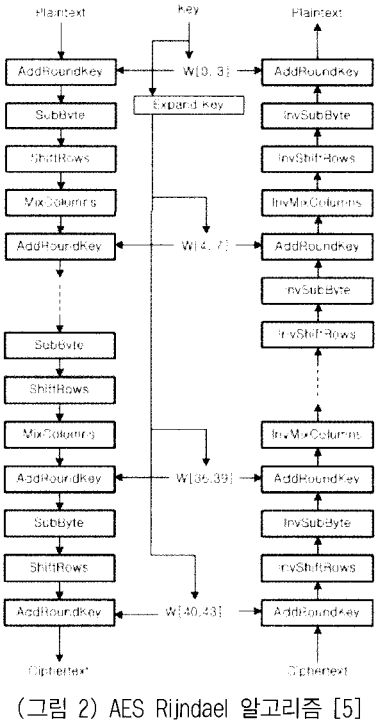
인터럽트란 업무 중인 사람이 전화가 울리면 업무를 잠시 멈추고 전화를 받는 것처럼 어떠한 프로세스를 수행하는 도중에 그 일을 잠시 멈추고 CPU의 업무 처리를 위해 CPU의 주의를 끄는 방법을 말한다[2], [3], [4]. 인터럽트에는 타이머 인터럽트, 비디오 인터럽트 등 많은 종류가 있으며 키보드 인터럽트도 그 중 하나로서 인터럽트 번호는 9번이다. 즉, 키보드의 키가 눌리거나 혹은 떼어질 때마다 이 인터럽트가 발생한다.

물리적인 키보드 입력이 일어나면 키보드의 전기적인 신호가 키보드 라인을 따라 컴퓨터의 마더보드로 전달된다. 이 때 마더보드 상의 Intel 8259 Programmable Interrupt Controller(PIC) 칩에 의해 키보드 디바이스로부터의 입력 신호를 받아들여 IRQ(Interrupt Request)를 통해 CPU에게 해당 인터럽트가 발생했음을 알려주게 되며, CPU는 해당 인터럽트 핸들러를 호출하기 위해 메모리에 적재되어 있는 인터럽트 벡터 테이블(IVT)를 읽어 해당 테이블에 기록되어 있는 주소값을 얻는다. 이 주소값은 인터럽트 발생시 호출되는 함수(인터럽트 핸들러)의 주소값을 나타내는 것으로서 디바이스가 프로세서에게 어떠한 연산이나 일을 수행하게 할 때 이 주소의 위치로 이동하여 핸들러를 수행하게 된다[2], [3]. 그러므로 인터럽트 벡터 테이블에 기록되어 있는 인터럽트 핸들러가 디바이스에 대한 연산을 처리하는 과정에서 볼 때 가장 먼저 수행되는 것이라 볼 수 있다.

2.3 AES Rijndael 암호화 알고리즘

DES가 더 이상 안전성을 보장할 수 없게 되자, NIST에 의해 새로운 AES로 Rijndael이 채택되었다. Rijndael 암호 알고리즘은 알려진 모든 공격에 강하고, 그 응용에 있어서 속도나 하드웨어 구현에 뛰어난 장점을 가지고 있다.[5][6]

Rijndael의 구조는 (그림 2)와 같이 크게 4개의 연산으로 이루어진다. SubByte는 S-Box를 이용하여 byte 단위의 치환을 수행하며, ShiftRows는 열의 순서를 치환시킨다. MixColumn은 GF(2^8) 연산을 이용한 치환이다. AddRoundKey는 bit-wise-XOR 연산을 이용하여 Round Key를 생성한다.



(그림 2) AES Rijndael 알고리즘 [5]

3. 시뮬레이션 설계

본 논문의 시뮬레이션에서는 전자상거래를 이용하는 사용자의 키보드 입력이 웹서버로까지 안전하게 전달되도록 하는 키보드 입력 보안 시스템을 제시한다. 그리고 그러한 보안 시스템을 이용하여 사용자의 아이디와 패스워드를 입력한 후 확인 버튼을 눌렀을 때, 정상적인 아이디와 패스워드를 입력했는지를 처리하는 간단한 사용자 인증 절차를 구현하였다.

3.1 모듈 설명

본 시스템은 크게 키보드 입력값을 암호화하는 키보드 드라이버 및 키보드 인터럽트 핸들러와 이 값을 받아서 때에 따라서 복호화를 수행하고 HTML의 은닉 필드(hidden field)에 암호화된 값을 넣어주는 역할을 수행하는 ActiveX 형태의 브라우저 임베딩 모듈, 그리고 웹서버 내부에서 이 값을 복호화하여 처리하는 서버 컴포넌트 이렇게 세 가지 모듈로 구성되어 있다.

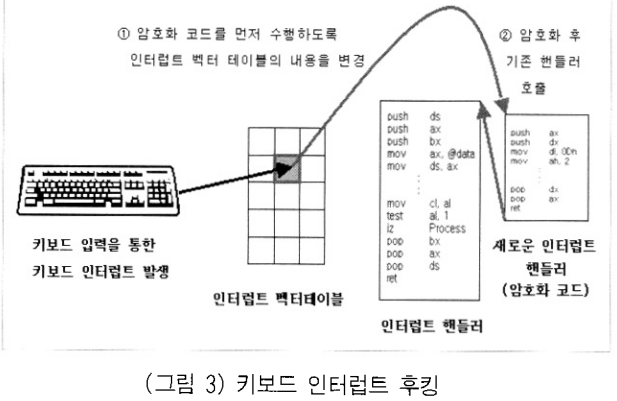
모든 모듈은 Visual C++환경에서 제작되었으며, 키보드 드라이버의 경우 윈도우 95/98/ME를 위해 vxd의 형태로, 윈도우 NT/2000/XP는 sys형태로 제작되었다.[7][8]

3.1.1 키보드 드라이버 및 키보드 인터럽트 핸들러

키보드 인터럽트 핸들러를 설치하기 위해 키보드 드라이버가 필요하며, 본 시스템에서는 키보드 상위 필터 드라이버(upper filter driver)를 개발하여 키보드 인터럽트 후킹을 수행하도록 하였다.

키보드 인터럽트 후킹(Keyboard Interrupt Hooking)이란 (그림 3)과 같이 인터럽트 벡터 테이블에 기록되어 있는 인터럽트 핸들러의 주소를 다른 핸들러의 주소로 바꾸어놓는 기술을 말한다. 키보드 인터럽트 핸들러는 키보드 인터럽트가 발생했을 때 호출되는 일종의 함수를 말하는 것으로, 키보드 입력에 대한 처리를 담당한다. 본 시스템에서는 기존의 키보드 인터럽트 핸들러가 실행되기 전에 자체적으로 제작된 키보드 인터럽트 핸들러가 먼저 실행되게끔 키보드 인터럽트를 후킹한다.

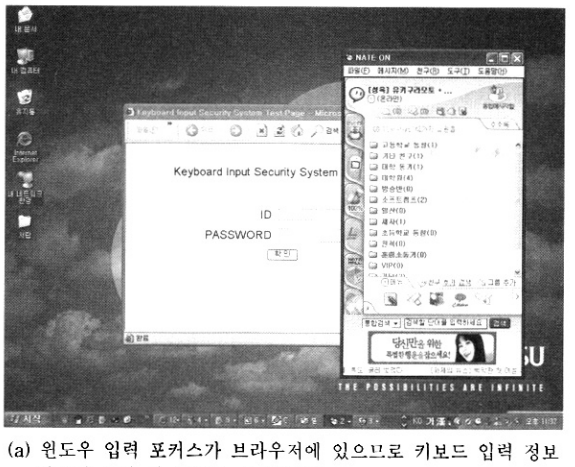
또한 키보드 드라이버는 다른 프로그램이 키보드 값을 제일 먼저 가져갈 수 없게끔 주기적으로 인터럽트 벡터테이블을 검사하여 자체 제작된 키보드 인터럽트 핸들러가 키보드 입력값을 가장 먼저 가져갈 수 있게끔 인터럽트 벡터테이블에 자체 제작된 키보드 인터럽트 핸들러의 주소가 기록되어 있는지 확인하고, 기록되어 있지 않으면 그 즉시 기록한다.



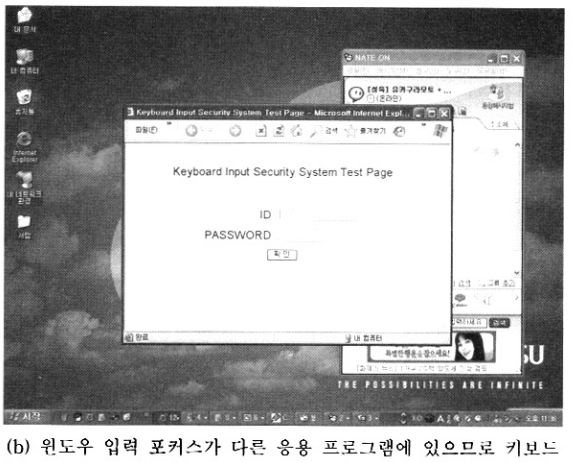
(그림 3) 키보드 인터럽트 후킹

3.1.2 브라우저 임베딩 프로그램

인터넷 익스플로러(Internet Explorer)와 같은 웹브라우저 내에서 동작하는 프로그램으로서 키보드 입력 보안 시스템의 메인 프로그램으로서 ActiveX 형태의 입력 컨트롤로서 일반적인 HTML의 입력 태그(<input>)와 동일한 형태를 띤다. 사용자는 본 컨트롤을 통해 키보드를 입력하게 된다.



(a) 윈도우 입력 포커스가 브라우저에 있으므로 키보드 입력 정보 암호화 동작 및 암호화 키 재분배



(b) 윈도우 입력 포커스가 다른 응용 프로그램에 있으므로 키보드 입력 정보 암호화 중지

(그림 4) 윈도우의 입력 포커스에 따른 키보드 입력 정보 암호화 동작 여부

일반적으로 다른 응용프로그램이나 브라우저 임베딩 프로그램이 웹페이지에 접근하게 되면 HTML의 입력 태그(<input>)의 저장내용을 알 수 있기 때문에, 접근하더라도 화면에 저장되어 있는 키보드 입력값을 가져가지 못하도록 하기 위해 본 모듈은 자체적인 입력 컨트롤 형태로 제작되었다.

본 모듈은 내부적으로 최초 동작 시에 키보드 드라이버를

설치 및 암호화 키를 재분배하는 역할을 수행하며 또한 현재 윈도우의 입력 포커스가 다른 응용프로그램에게로 가게 되면 키보드 입력 정보 암호화를 수행하지 않도록 하고, 다시 입력 포커스가 키보드 입력 보안 프로그램이 동작하고 있는 웹브라우저에 오게 되면 새롭게 암호화 키를 재분배하고 키보드 입력 정보 암호화를 수행할 수 있도록 조정한다.

또한 키보드 인터럽트 핸들러로부터 받은 암호화된 키보드 입력값 중에서 텍스트 필드상의 입력값은 복호화하여 화면에 출력하고 패스워드 필드의 입력값은 복호화하지 않고 단순히 *와 같은 별표(asterisk)를 찍어주는 역할을 수행한다. 그리고 최종적으로 암호화된 값은 UUENCODE 기법을 통해 텍스트로 변환하여 HTML상의 은닉 필드의 값으로 저장한다.

3.1.3 서버 컴포넌트

서버 컴포넌트는 웹서버 내에서 구동되는 모듈로서 사용자의 키보드 입력이 일어난 클라이언트 PC에서 서버로의 제출(submit)이 일어났을 때 암호화되어 있는 키보드 입력값이 들어 있는 은닉 필드의 값을 네트웍을 통해 받아서 이를 복호화함으로써 사용자가 입력한 원래의 키보드 입력값으로 복원하여 처리하는 역할을 수행한다.

3.2 키보드 입력 보안 시스템 동작 절차

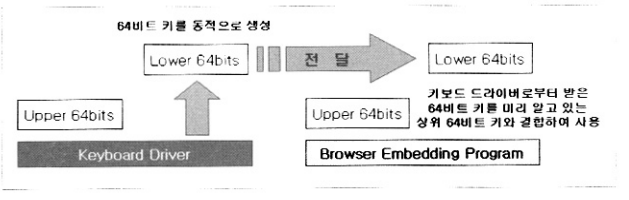
본 시뮬레이션의 핵심은 본 프로그램이 구동되는 인터넷 익스플로러(Internet Explorer) 브라우저 상에서 키보드 입력이 일어나는 순간에 소프트웨어적으로 가장 먼저 발생하는 키보드 인터럽트 단에서 해당 키보드 입력 내용을 암호화하여 네트웍 구간을 거쳐 웹서버로 전달한 후 최종적인 웹서버단에서 값을 복호화하여 처리하자는 것이다.

3.2.1 키보드 인터럽트 후킹 및 암·복호화 키 분배

최초에 프로그램이 동작하면, 브라우저 임베딩 프로그램은 키보드 드라이버를 설치하고, 키보드 드라이버는 키보드 인터럽트를 후킹하여 암호화를 수행하는 자체 키보드 인터럽트 핸들러를 인터럽트 벡터 테이블에 연결시켜 놓는다. 그리고나서 키보드 드라이버는 난수 발생기(Random Number Generator)를 이용하여 암·복호화 키를 생성하고 이 값을 브라우저 임베딩 프로그램에게 전달한다. 이 때 128비트 키 전체를 동적으로 생성하는 것이 아니라, 키보드 드라이버와 브라우저 임베딩 프로그램 모두 코드 내에 정적으로 동일한 64비트 비밀키를 가지고 있기 때문에 나머지 64비트만 동적으로 생성하게 된다. 이렇게 64비트 비밀키를 미리 동일하게 갖도록 설계한 이유는 암·복호화 키 교환시에 어떠한 식으로든 그 키 값을 가로채는 것을 대비하기 위함이다.

만약 윈도우의 입력 포커스가 다른 응용프로그램에게 넘어가면 브라우저 임베딩 프로그램은 키보드 드라이버에게 키보드 인터럽트 후킹을 해제하도록 명령하여 키보드 입력 정보 암호화를 중지한다. 또한 윈도우의 입력 포커스가 다시 브라우저 임베딩 프로그램이 동작하는 브라우저로 돌아오게 되면 브라우저 임베딩 프로그램은 다시 키보드 드라이버에게 키보드 인터럽트 후킹을 명령하고 암호화 키를 새로이 생성하여 재분배함으로써 키보드 입력 정보 암호화를 시작하게 된다.

또한, 주기적으로 키보드 드라이버는 키보드 인터럽트 벡터 테이블의 주소가 자신의 키보드 인터럽트 핸들러의 주소인지 확인하고 아닐 경우에 즉시 자신의 핸들러 주소로 바꿔놓음으로써 다른 응용 프로그램보다 가장 먼저 키보드 입력값을 가져올 수 있게끔 한다.



(그림 5) 암·복호화 키 분배 방식

3.2.2 키보드 입력 정보 암호화 수행

키보드 인터럽트 후킹을 하게 되면 새로운 키보드 인터럽트 핸들러는 사용자가 입력한 키보드 입력값을 암호화하게 되는데, 암호화 알고리즘은 128비트 AES Rijndael 알고리즘을 사용하였으며, 키보드 입력값은 1바이트이기 때문에 128비트, 즉 16바이트를 모두 0으로 초기화한 후 임의의 위치에 입력값 1바이트를 넣고 암호화를 수행한다. 이 때 임의의 위치에 값을 넣는 이유는 같은 키의 키보드 입력 값이라도 매번 암호화된 결과가 다르게 나타나도록 만듦으로써 반복되는 암호화 데이터 패턴을 이용한 원본 데이터 추측 시도를 대비하기 위함이다. 암호화 모듈은 키보드 입력 값을 암호화한 후 암호화된 값을 브라우저 임베딩 모듈에게 직접 통신을 통해 전달하며, 다른 응용프로그램들이 키보드 입력값을 알아낼 수 없도록 입력값을 지운 후 원래의 키보드 인터럽트 핸들러로 리턴한다.

3.2.3 브라우저 화면에 키보드 입력값 출력

앞서 3.1.2절에서 설명한 바와 같이 키보드 입력이 일어났을 때 HTML 상의 일반적인 텍스트 필드와 같이 입력값을 그대로 화면에 출력해야 하는 경우가 발생하므로 이 때에는 복호화하는 것이 불가피하다. 그러나 이는 화면에 보여지는 값에 한해서이고, 패스워드 필드와 같이 *로 표시되는 문자는 중간에 복호화하지 않고 키보드 입력시마다 화면에 단순히 *를 출력하고, 실제 입력값은 암호화된 채로 그대로 서버에 전송되게 된다.

또한 복호화하는 텍스트 필드의 경우에도 화면에 보여지기 위해 잠시 메모리 상에서 복호화할 뿐, 실제 은닉 필드를 통해 서버로 데이터를 넘길 때는 *로 표시되는 패스워드와 동일하게 암호화된 데이터를 전송하게 된다. 또한 화면에 보이는 컨트롤 자체도 접근 방법이 알려져 있는 일반적인 HTML의 태그와 달리 복호화된 값을 가져오는 인터페이스가 없는 보안 컨트롤이므로 위험하지 않다.

3.2.4 은닉 필드를 통한 서버로의 입력값 전달

실제 서버로의 제출(submit)이 일어나면, 자바스크립트 함수 호출을 통해 브라우저 임베딩 프로그램은 자신의 입력 컨트롤이 가지고 있는 암호화된 키보드 입력값을 HTML의 각 은닉 필드에 저장하게 되고, 실제 <action>태그의 POST 또는 GET방식을 통해 서버에 전송된다.

```
<input type="hidden" name="h_id">
<input type="hidden" name="h_password">
```

(그림 6) 은닉 필드의 설정

이 때 중요한 것은 동적으로 생성된 64비트 암·복호화 키도 함께 은닉 필드를 통해 서버로 전달되어야 한다는 것이다. 물론 서버 컴포넌트에도 브라우저 임베딩 프로그램이나 키보드 드라이버와 동일하게 정적으로 생성되어 있는 동일한 64비트 비밀키를 기존에 가지고 있다. 또한 HTTP 프로토콜의 특성상 전달 데이터는 텍스트이어야 하므로 UUENCODE 방식을 이용하여 이진 데이터로 되어 있는 암호화된 키보드 입력값과 동적으로 생성된 64비트 암·복호화 키를 아스키 형태로 인코딩하여 은닉 필드에 저장해야 한다.

물론 이러한 전송 데이터는 불특정 다수가 가로챌 수 있는 위험성을 띤 네트웍 구간을 지나가게 되지만, 데이터 자체가 암호화되어 있으며, 또한 PKI나 SSL을 통한 네트웍 구간에 대한 보안 시스템은 이미 많은 전자상거래 사이트에 도입되어 있으므로, 본 논문에서 네트웍 구간에 대한 보안은 주요한 관심사가 아님을 밝힌다.

3.2.5 서버 컴포넌트 상의 입력값 복호화

은닉 필드의 데이터를 받은 웹서버의 서버 컴포넌트는 먼저 이 값을 UUDECODE 방식을 통해 이진 데이터로 디코딩한 후 전달받은 64비트 암·복호화 키와 코드 내에 정적으로 가지고 있는 64비트 비밀키를 조합하여 AES Rijndael 알고리즘을 통해 암호화되어 있는 키보드 입력값을 복호화한다. 이 때 복호화된 키보드 입력 한글자를 위한 16바이트 암호화 데이터 값 중에서 0으로 설정되지 않은 값이 실제 입력값이므로 0이 아닌 데이터를 이용한다. 이와 같이, 같은 키보드 입력에 대해서도 실제 암호화된 데이터가 각각 다르므로 암호화 데이터를 실제 복호화하지 않으면 원래 데이터를 알아낼 수 없다. 그리고 최종적으로 이렇게 복호화된 키보드 입력 데이터를 이용하여 사용자 인증 등 일반적인 처리를 수행한다.

4. 실험 결과 및 분석

4.1 사용자 입력 정보의 침해 유형에 따른 각 구간별 해킹 시도

키보드 입력이 일어나는 순간부터 서버로 전송되기까지 소프트웨어적으로 사용자의 입력 정보를 가로챌 수 있는 해킹 시도가 가능한 방식은 크게 키보드 인터럽트 후킹 단계, 키보드 필터 드라이버 단계, API 후킹을 통한 데이터 가로채기 단계, 윈도우 메시지 후킹 단계, 브라우저 접근 단계, 그리고 네트웍 패킷스니핑 단계의 총 6가지 형태로 나타낼 수 있다.

먼저 키보드 인터럽트 후킹 단계는 사용자가 임의로 키보드 인터럽트 핸들러를 작성하여 기존의 핸들러보다 먼저 키보드 값을 가져갈 수 있게끔 가로채는 방식이다. 본 방식은 앞서 설명한 본 시스템의 인터럽트 핸들러와 동일한 방식으로서 새로운 함수를 작성하여 인터럽트 벡터 테이블의 주소를 자신의 함수 포인터로 바꿔놓는 방식으로서 도스 운영체제 시절, 게임 프로그래밍 등에 많이 사용되었던 방식이다.

두 번째의 키보드 필터 드라이버 공격 방식은 운영체제의 디바이스 드라이버 키트(DDK)와 같은 소프트웨어 개발 도구(SDK)를 이용하여 키보드 디바이스 드라이버를 작성하여 등록하는 방식으로서 윈도우의 기본 키보드 디바이스 드라이버인 kbdclass와 같은 방식의 상위 또는 하위 필터 드라이버를 개발하여 등록함으로써 키보드 값을 얻어오는 방식이다. 보통 특수한 키버튼이 존재하는 특수 키보드 개발 업체에서는 본 드라이버를 작성하게 된다. 이러한 디바이스 드라이버는 순차적으로 연결되어 있으므로, 상위 드라이버부터 키보드의 값을 차례로 얻어오게 된다.

세 번째 방식은 윈도우 메시지 후킹 방식이다. 시스템 영역의 키보드 디바이스 드라이버는 응용 프로그램 영역으로 키보드 입력값을 넘겨주기 위해 윈도우 운영체제가 관리하는 메시지 큐에 키보드 입력값을 WM_KEYDOWN 등의 윈도우 메시지로 변환하여 넣어두게 되며, 윈도우 상의 각종 응용 프로그램들은 이러한 윈도우 메시지를 받아서 메시지 기반(Message Driven) 방식으로 프로그램을 운영한다. 이 때 윈도우 메시지 후킹이란 별도의 응용 프로그램을 작성하여 자신에게 발생한 이벤트가 아닌 다른 윈도우에서 발생한 윈도우 이벤트까지 가로채는 방식을 말한다.

네 번째 방식은 API 후킹 방식이다. API 후킹도 윈도우 메시지 후킹 방식과 비슷한 방식으로서 특정 API가 실행될 때 그 함수가 실행되기 전에 가로채어 사용자가 작성한 함수를 먼저 실행하는 것을 말한다. 키보드 보안 시스템이 디바이스 드라이버와 윈도우의 브라우저 내의 ActiveX가 서로 통신하는 형태이기 때문에, 이 때 사용하는 통신 API인 DeviceIoControl() API를 후킹하여 함수의 파라미터인 암호화 데이터를 알아내는 방식이다.

다섯 번째 방식은 인터넷 익스플로러 브라우저에 COM 인터페이스를 통해 접근하는 방식이다. 이러한 방식도 세부적으로 브라우저 외부 응용프로그램이 접근하는 형태와 브라우저 내부의 브라우저 헬퍼 오프젝트(Browser Helper Object)가 접근하는 형태로 나눌 수 있다. 이 방식을 이용하면 IWebBrowser 등의 COM 인터페이스를 통해 현재 브라우저가 보여주고 있는 화면의 특정 화면 텍스트 데이터나 HTML 소스, 사용자가 입력한 데이터가 저장되어 있는 HTML 입력 필드 내 데이터 등에 접근이 가능하다.

〈표 1〉 사용자 입력 정보의 침해 유형과 각 구간별 해킹 시도 결과
(키보드 보안 모듈 설치 후 "internet"이라는 8글자를 입력)

침해 유형	해킹 방법	본 시스템 적용 결과	방식 설명
키보드 인터럽트 후킹	별도의 인터럽트 후킹 드라이버를 자체 제작하여 후킹 시도	안전 (키보드 입력값 가로채기 실패) * 가로챈 값 : 없음 (NULL)	입력이 일어나는 순간 혹은 주기적으로 인터럽트 벡터 테이블을 검사하여 인터럽트 핸들러 재등록
키보드 필터 드라이버 공격	다음의 알려진 툴을 이용 : IKS / Invisible Keylogger 97 / KeyKey 2002 Pro / XSoftware	안전 (키보드 입력값 가로채기 실패) * 가로챈 값 : 없음 (NULL)	키보드 인터럽트 단에서 미리 값을 읽어서 암호화한 후 값을 지우므로 키보드 드라이버 단에서는 키보드가 입력된 사실을 알지 못함
API 후킹	DeviceIoControl() API 후킹을 통한 데이터 가로채기	안전 (암호화된 데이터) * 가로챈 값 : 암호화된 데이터	매번 암호화 시마다 같은 키라도 다른 암호화 값이 생성되므로 추측 불가 (원천적인 API후킹방지는 시스템의 안정성을 위해 구현하지 않았음)
윈도우 메시지 후킹	다음의 알려진 툴을 이용 : SPY++/ 2Spy / 백오리피스2K / Ghost Keylogger / KeyInterceptor / SC-KeyLog2	안전 (키보드가 입력됐다는 메시지만 가로챌 수 있으며, 어떤 키가 눌려졌는지는 알지 못함) * 가로챈 값 : 없음 (NULL)	브라우저 임베딩 프로그램에서 인터럽트 핸들러로부터 직접 값을 받아서 복호화하므로 윈도우 메시지는 단순한 타이밍 역할만 할 뿐, 복호화시 사용하지 않음
브라우저 접근	BHO등을 통해 인터넷 익스플로러 브라우저의 COM인터페이스를 통한 입력 컨트롤 직접 접근 시도 다음의 알려진 툴을 이용 : Perfect Keylogger	안전 (이미 알려진 인터넷 익스플로러의 COM 인터페이스와는 관계 없으며, 은닉 필드의 경우 제출(submit) 순간에 잠시 암호화된 값만을 가져갈 수 있음) * 은닉필드 가로채기 값 : 172바이트 데이터1)	HTML의 입력 태그의 필드에 대한 접근이 가능하지만, 본 시스템의 입력 컨트롤은 자체 제작된 것이므로 안전하며, *로 나타나는 패스워드 필드의 경우 웹서버에 도달할 때까지 복호화하지 않으므로 더욱 안전함
네트워 상의 스니핑 공격	네트워 패킷 스니퍼를 통한 전송 데이터 가로채기 시도	안전 (UUENCODE로 변환된 암호화 데이터) * 가로채기 값 : 172바이트 데이터2)	암호화되어 있는 값을 가로채게 되므로 안전

〈표 2〉 타 키보드 입력 보안 시스템과의 비교

비교 항목	본 시스템	Secure KeyStroke	Anti-Keylog
키보드 인터럽트 구간	안전 (가로채기 감지)	안전 (가로채기 감지)	안전 (가로채기 감지)
키보드 드라이버 구간	안전 (가로채기 불가)	안전 (가로채기 불가)	안전 (가로채기 불가)
API 후킹	안전 (반복되지 않는 암호화 데이터만을 가로채기 때문에 값 추측 불가)	안전 (API Hooking방지)	안전하지 못함 (암호화데이터를 가로채게 되지만 반복되는 패턴을 통한 값 추측 가능)
윈도우 메시지 가로채기	안전 (가로채기 불가)	안전 (가로채기 불가)	안전 (가로채기 불가)
브라우저 접근	안전 (패스워드의 경우 복호화 자체를 하지 않으며, 아이디의 경우 화면상에 복호화되어 있는 값이 보이지만, HTML태그와는 다른 보안컨트롤이므로 그 값에 접근할 수 없다.)	안전하지 못함 (암호화구간이 아니기 때문에 복호화가 일어나며, 복호화된 값을 해킹 프로그램이 가로챌 위험이 존재함)	안전하지 못함 (암호화구간이 아니기 때문에 복호화가 일어나며, 복호화된 값을 해킹 프로그램이 가로챌 위험이 존재함)
네트워 스니핑	안전 (암호화된 데이터를 가로채게 된다.)	안전하지 못함 (암호화구간이 아니기 때문에 반드시 PKI나 SSL과 같은 네트워 보안 모듈을 탑재해야 함)	안전하지 못함 (암호화구간이 아니기 때문에 반드시 PKI나 SSL과 같은 네트워 보안 모듈을 탑재해야 함)

1), 2) : 암호화 데이터량이 172바이트가 된 이유는 8자의 입력글자에 대한 AES암호화 데이터인 128바이트가 UUENCODE 기법을 통해 텍스트로 변환되는 과정에서 데이터가 늘어났기 때문임.

마지막으로 네트워크 패킷 스니핑 방식이다. 이 방식은 브라우저에서 제출(submit) 등으로 실제 데이터가 웹서버로 전달되는 순간에 네트웍 구간의 데이터를 가로채어 내용을 몰래 엿보는 방법이다.

위에서 설명한 각 구간별로 본 키보드 입력 보안 시스템에 대한 해킹 시도의 결과를 다음의 <표 1>에 나타내었다. 아래에서도 볼 수 있듯이 키보드 입력 내용이 웹서버로 전달되기까지의 전 구간이 안전하다는 것을 볼 수 있다.

4.2 기존 시스템과의 비교

본 논문에서 제안하는 키보드 입력 보안 시스템을 기존에 개발된 타 키보드 입력 보안 시스템들과 비교한 것을 <표 2>에 나타내었다. 물론 기존의 키보드 입력 보안 시스템들은 순수하게 키보드 입력 후 화면에 출력되기 이 전까지를 보안 구간으로 설정한 것이 제품의 목적이기 때문에 네트웍 구간 등 기타 브라우저 화면으로의 키보드 값 출력 이 후의 구간은 암호화 영역으로 포함하지 않는다. 그러므로 본 논문에서의 키보드 입력 보안 시스템은 기존 보안 시스템의 확장판이라 할 수 있다.

최근 키보드 입력에 대한 보안 이슈가 대두되면서 금융권을 중심으로 키보드 입력 보안 솔루션을 도입하고 있으나 기존에 구축된 PKI 솔루션에 추가적인 형태로 구축이 되다 보니 브라우저 상에서 한번 복호화하는 과정이 생기게 되며, 위의 <표 2>와 같이 브라우저 접근에 대한 보안대책이 필요한 상태이며, 이는 키보드 보안 제품이 단독으로 구축되지 않는 이상 PKI 솔루션업체와 함께 풀어가야 할 숙제이다.

본 논문에서 제시한 시스템을 구축하기 위해서는 각 입력 필드마다 오브젝트 태그(<object>) 코드를 삽입해야 하고 데이터 전송을 위해 입력 필드를 설정해야 하는 번거로움은 있으나, BHO 등의 브라우저 접근 공격 측면에서 기존의 시스템보다 더 높은 보안성을 제공하며, 또한 클라이언트 PC 상에서 *로 출력되는 중요 데이터를 복호화하는 과정을 없앰으로써 메모리 스캔 등으로 인한 보안의 홀이 발생할 수 있는 여지를 제거하였다.

5. 결론 및 향후 연구과제

본 논문에서는 전자상거래 상에서의 더욱 안전하고 신뢰성있는 키보드 입력 보안 시스템을 제안하였다. 최근 8051 마이크로 컨트롤러를 이용한 하드웨어 기반의 키보드 보안 시스템도 등장하고 있으나[9], 배포의 어려움이나 가격적인 부담으로 인해 인터넷 뱅킹과 같은 대고객 서비스를 위한 솔루션으로는 적합지 못하다. 본 보안 시스템은 순수 소프트웨어 방식이므로 ActiveX를 통한 손쉬운 배포가 가능할 뿐만 아니라 어떠한 해킹툴이 새로이 출현하더라도 키보드의 값을 근본적으로 보호하기 때문에, 바이러스 백신 등과 다르게 주기적인 업데이트나 엔진교체가 필요없다는 장점을 가진다. 또한 10여가지의 상용툴을 이용하여 키보드 입력값이 전달되는 여러 구간마다 해킹 가능 여부를 테스트하였으며, 이에 안전하다는 것을 입증하였다. 본 시스템은, 무엇보다도 기존 시스템에 비해 브라우저 공격에 강하고, 클라이언트 PC상에서의 복호화 과정을 줄임으로써 메모리 스캔 등으로 인한 보안의 홀이 생길 수 있는 위험을 제거함으로써 더 높은 보안성을 가진다는 개선점을 지니고 있다.

그러나 현재 본 시스템은, 보안성과 사용의 편리성은 상호 절충(trade-off)의 관계라는 일반적인 통념을 보여주듯 구축이 다소 까다롭다는 단점을 가지고 있기 때문에 용이한 구축 방안에 대한 연구가 필요하다. 또한 본 시스템은 윈도우 운영체제에서만 동작한다는 기술적인 한계점을 지니고 있다. 이 한계점을 극복하기 위해, 리눅스의 경우 LKM (Linux Kernel Module)기법을 통한 키로거를 이용하는 방안을 생각해 볼 수 있으며[10], 기타 시스템에 대해서도 각 시스템마다 적용할 수 있는 시스템 프로그래밍 기법에 대한 연구가 필요하다. 이 두 가지 부분만 개선된다면 기존의 보안 시스템을 대체하여 폭넓게 사용될 수 있을 것이다. 따라서, 향후 연구 과제로서 이러한 문제점을 개선한 키보드 입력 보안 시스템 개발 방법에 대한 연구가 필요하다.

참 고 문 헌

[1] 이현우, "네트워크 공격기법의 패러다임 변화와 대응방안", pp.10, SecurityMap, 2001.

[2] 김석주, "C로 하드웨어 주무르기", 1st Ed., pp.117~174, pp.55~59, 가메출판사, 1999.

[3] N. Barkakati, R. Hyde, "매크로 어셈블러 바이블 한국어판", 1st Ed., pp.36~38, pp.695~712, 인포북, 1993.

[4] 이동기, 김태한, "C에서 어셈블러까지", 2nd Ed., pp.67~78, 월간 PC어드밴스 단행본부, 1995.

[5] T. Jamil, "The Rijndael algorithm", IEEE Potentials, Vol.23, No.2, pp.36~38, Apr., 2004.

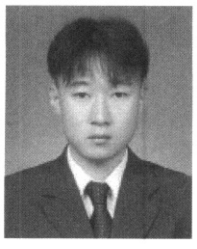
[6] W. Stallings, "Cryptography and Network Security: Principle and Practice", 3rd Ed., Prentice Hall, 2003.

[7] K. Hazzah, "Writing Windows VxDs and Device Drivers", 2nd Ed., R&D Books, 1997.

[8] E. Dekker, J. Newcomer, "Developing Windows NT Device Drivers: a programmer's handbook", 1st Ed., Addison Wesley, 1999.

[9] D.G. Treat, "Keyboard encryption", IEEE Potentials, Vol.21, No.3, pp.40~42, Aug./Sep., 2002.

[10] 이상인, 박재홍, 강홍식, "허니팟을 위한 원격 키스트로크 모니터링의 설계", 정보과학회 춘계학술대회, Vol.31, No.2, pp.367~369, Oct., 2004.



최 성 욱

e-mail : csucom@paran.com

1999년 중앙대학교 컴퓨터 공학과(학사)

2001년 중앙대학교 대학원 컴퓨터 공학과 (석사)

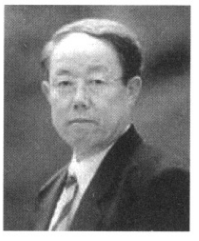
2003년 중앙대학교 대학원 컴퓨터 공학과 박사 수료(박사과정)

2003년 중앙대학교 공과대학 컴퓨터공학과 강사

2003년 서울여자대학교 정보통신공학부 겸임교수

1999년~현재 소프트캠프(주) 정보보안기술연구소 선임연구원

관심분야 : 인공생명, 인공지능, 게임 S/W, 그래픽스, 인터넷 보안 등



김 기 태

e-mail : ktkim@ailab.cse.cau.ac.kr

1966년~1972년 중앙대학교 공과대학 조교

1971년~1972년 중앙대학교 전자계산소 조교

1972년~1979년 중앙대학교, 한양대학교, 홍익대학교, 아주대학교 강사

1979년~2005년 중앙대학교 공과대학 전자계산학과 교수

1988년~1990년 중앙대학교 공과대학 교학부장

1991년~1992년 Connecticut State Univ. 교환교수

1997년~1999년 중앙대학교 정보산업대학원 원장

1997년~1999년 중앙대학교 전산정보처 처장

1997년~1999년 중앙대학교 수퍼컴연구소 소장

현재 중앙대학교 공과대학 컴퓨터공학부 명예교수

관심분야 : 인공지능, 인공생명, CBR, 신경망 등