USB 저장 장치와 스마트폰을 이용한 윈도우 OS PC 보안 시스템 구현

이세훈*, 김지성O,탁진현**
*O인하공업전문대학 컴퓨터시스템과
**신안산대학교
e-mail : seihoon@inhatc.ac.kr*, succubus93@naver.comO, tak@sau.ac.kr**

Implementation of Windows OS PC Security System using USB Storage Device and Smartphone

Se-Hoon Lee*, Ji-Seong KimO,Jin-Hyun Tak**
O*Dept. of Computer Systems & Engineering, Inha Technical College
**ShinAnSan University

● 요 약 ●

본 논문에서는 PC가 사용자를 구분하지 못하여 발생하는 보안 문제점과 개인이 PC를 관리하는 시간이 증가함에 따라 관리자가 미처 신경 쓰지 못하게 되는 보안 문제를 해결하고자 USB 저장 장치와 스마트폰을 이용한 사용자 PC인증 및 PC환경에서의 물리적 외부 접근 차단, 기업 차원이 아닌 개인 차원에서의 저 비용 보안 기술을 적용한 시스템을 구현하였다.

키워드: PC 보안(PC Security), SEED 암호화 알고리즘(SEED Encryption Algorithm), Win32API

I. 서 론

대부분의 정보처리 업무가 PC기반으로 이루어지며 PC의 세대별 하드웨어 성능 증가 폭이 계속 줄어듦에 따라 기존의 PC의 기대수명 또한 증가하게 되었다. 이로 인해 PC 보급률 또한 증가하였으며 이 결과로 대부분의 사용자가 업무용 PC와 개인용 PC를 따로 사용함에 따라 사용자가 총괄적으로 PC를 관리하는 시간이 증가하게 되었다. 그러므로 개인이 관리하는 PC의 보안에 대한 책임 또한 늘어나게 되었는데, 상당수의 정보유출이 내부자에 의해 발생하는 만큼 개인의 PC보안 관리의 중요성이 대두된다. [1]에 따르면 알파벳-숫자 기반의 기억해야할 정보의 종류가 결정된 패스워드는 입력수단의 버튼을 누르는 행위는 Shoulder-surfing 에 취약하며, 화면에 패스워드 대신 '*' , '●' 특수문자를 표현하는 것은 공격자가 입력장치를 바라보는 행위에 대해서는 아무런 해결책이 될 수 없다. 따라서 본 논문에서는 개인의 PC보안 허점을 해결하기 위해 USB 저장장치 또는 스마트폰을 개인 인증도구로 이용하여 원치 않은 사용자의 PC접근을 제한하는 반 물리적 보안시스템을 구현하고자 한다.

II. 관련 연구 고찰

1. SEED암호화 알고리즘

스마트폰에 의한 PC 보안을 하려할 때 UDP통신을 이용하여 지정된 ID를 전송하게 되는데, 이 때 보안되지 않은 평문을 전송한다면 제3자가 패킷을 스니핑하여 패킷 재전송 공격을 할 수 있는 문제점이 있다. 이러한 문제 해결을 위해 [2][3]를 따르면 블록암호 알고리즘 SEED-CBC는 평문 한 글자만 바뀌어도 전체 암호문의 내용이 가변한다. 그러므로 이러한 특성을 이용하여 패킷 재전송 공격을 방지하기 위해 평문에 타임스탬프를 혼합하여 암호문을 생성해 이를 전송한다. 그림1 은 암호화되지 않은 평문을 전송했을 때의 패킷분석 결과이다. ID가 변하지 않으며 재 전송 공격 시 보안이 바로 무력화 될 수 있다. 그림 2는 앞에서의 그림 1의 문제점을 보완하여 ID를 타임스탬프와 혼합한 후 KISA SEED - CBC 암호화 알고리즘으로 암호화한 후의 결과이다. 매 번 실행 시 마다 ID가 가변하여 제 3자의 패킷 관찰로부터 ID를 보호한다.

```
TYPE=LINK:
UNLOCK
TYPE=LOCK:ID=62711933164:
LOCK
TYPE=LOCK:ID=62711933164:
UNLOCK
TYPE=LINK:
UNLOCK
TYPE=LOCK:ID=62711933164:
LOCK
TYPE=LOCK:ID=62711933164:
UNLOCK
```

Fig. 1. Packet Analysis - Non Encryption

```
TYPE=LINK:
UNLOCK
TYPE=LOCK:ID=41,41,153,136,207,189,123,70,233,141,123,0,141,12,85,113,
127,178,208,152,35,175,150,234,136,176,182,171,14,130,17,222,:
LOCK
TYPE=LOCK:ID=134,121,204,226,65,93,134,24,212,233,180,109,210,15,3,30,
73,210,24,60,84,86,203,182,97,131,185,166,117,124,56,97,:
UNLOCK
TYPE=LINK:
UNLOCK
TYPE=LOCK:ID=80.220.118.242.208.204.51.47.154.93.1.164.232,60,213,74,
64,144,3,197,218,72,245,154,218,189,41,3,164,212,33,213,:
LOCK
TYPE=LOCK:ID=62,71,193,31,64,61,3,188,99,113,148,52,197,111,63,64,195,
105,0,251,251,153,48,28,201,168,15,208,49,29,14,209,:
UNLOCK
```

Fig. 2. Packet Analysis - Encryption

2. 작업관리자를 배제하기 위한 연구

외부 사용자로부터 PC를 보호하기 위해 본 보안시스템이 실행되어지고 있을 때 작업관리자(TaskManager)가 1순위로 실행된다면 본 시스템을 강제종료할 수 있기 때문에 작업관리자를 배제하도록 프로그램을 설계해야한다. 또한 작업표시줄(TaskBar)의 실행 권한이 관리자 권한보다 높으므로 작업표시줄에서 작업관리자를 호출했을 때 1순위로 호출되기 때문에 작업표시줄 또한 배제하여야 한다. 작업관리자 호출 핫키(ctrl+shft+esc 또는 ctrl+alt+delete)는 시스템에서 관여하는 시스템키기 때문에 일반적인 전역 후킹으로는 LockScreen에서 작업관리자 호출을 막을 수 없다. 그러므로 작업관리자를 호출하는 시스템Dll파일을 교체하거나 dll인젝션 기법을 사용해 작업관리자 호출부분을 무력화하여야 한다. 본 논문에서는 라이브러리 교체 기법을 제안한다.

2.1 Windows5.1(Windows XP)까지의 해결책

윈도우즈 초기화 단계에서 Winlogon.exe 응용프로그램이 msgina.dll(로그온 사용자 인터페이스 라이브러리) 동적라이브러리를 로드한다. 이후 작업관리자를 로드하는 이벤트가 발생한다면 msgina.dll라이브러리에서 로드한 키보드이벤트 메시지큐가 작업관리자 호출 핫키를 감지하고 msgina응용프로그램을 호출하여 작업관리자를 실행한다. 이러한 순서로 실행되는 작업관리자를 배제하기 위해 앞에서 제안한 것과 같이 msgina를 호출하는 msgina.dll라이브러리를 교체하는 방법을 사용한다. MSDN에서 제공하는 msgina.dll 인터페이스를 기초로 msgina를 호출하는 메서드를 수정 후 빌드하여 기존의 msgina.dll파일을 교체한다. 코드1은 msgina를 호출하는 메서드에서 ctrl+alt+del 이벤트가 발생하였을 때 작업관리자를 호출하지 않도록 호출 코드를 삭제한다.[4][5]

Algorithm 1 msgina.dll 의 작업관리자 호출부

```
Begin
int WINAPI WlxLoggedOnSAS(PVOID pWlxContext, DWORD dwSasType, PVOID
pReserved){
    HANDLE hMutex;
    if(dwSasType == WLX_SAS_TYPE_CTRL_ALT_DEL) {
        return WLX_SAS_ACTION_NONE;
    }
    return MsGina.WlxLoggedOnSAS(pWlxContext, dwSasType, pReserved);
End
```

2.2 Windows6 (Windows Vista)부터의 해결책

작업관리자와 작업관리자를 호출할 수 있는 작업표시줄은 기본적으로 Z Order가 최상위인 윈도우이다. LockScreen(그림5) 호출시 본 시스템의 UAC권한이 관리자레벨이 아닌 경우 작업관리자를 강제로 종료할 수 없으므로 본 프로젝트 윈도우의 Z Order를 최상위로 하여 작업관리자를 배제하며, 작업관리자를 호출할 수 있는 작업표시줄의 윈도우에 메시지를 보내 HIDE상태로 만들어 배제한다.[6]

Algorithm 2 작업표시줄숨김 및 Z Order 변경

```
Begin
void Lock(){
…중략
//테스크바 HIDE
ShowWindow(FindWindow(L"Shell_TrayWnd", NULL), SW_HIDE);
SetForegroundWindow(hWnd_Main);          //Z Order 변경
…중략
}
End
```

3. Client HeartBeat 기술

PC가 사용자의 주변 존재 유무를 확인할 방법으로 HeartBeat 신호를 보내는 방법을 사용한다. HeartBeat 기술이란 같은 네트워크 상의 내부 망에서 스마트폰 클라이언트가 일정 주기로 존재 신호를 보내는 방식을 말하며, 이것을 HeartBeat라고 명명한다. 그러므로 서버가 이 신호를 받은 시점을 기준으로 카운트하도록 한 다음, 설정해 둔 시간이 지나도록 다음 신호가 전송되지 않는다면 서버는 클라이언트가 AP를 벗어났다고 판단한다. 이에 따르는 주의점으로는 클라이언트가 HeartBeat 신호를 계속 전송하기 위해서는 스마트폰이 Sleep 모드와 DeepSleep 모드에 진입하더라도 HeartBeat신호를 전송할 수 있어야 한다. 또한 클라이언트가 강제로 Sleep모드에 진입하지 못하는 Immortal Service방식으로 설계할 경우 스마트폰의 배터리 과다소모 문제가 발생한다. 그러므로 이 문제들을 해결하기 위해서 안드로이드 PowerManager 권한을 얻는 방법 또는 안드로이드 스케줄러를 이용해야 한다. PowerManager를 통해 원하는 시간에 작업을 수행한 후 다시 sleep모드로 진입하여 배터리소모를 줄일 수 있는데, 이 방법은 안드로이드 서드파티에 가입이 되어있어야만 제어 권한을 받을 수 있다. 그러므로 안드로이드 스케줄러를 이용해 HeartBeat인터페이스를 구현해야한다. 위 사항에 따라 안드로이드 AlarmManager

를 이용하면 간접적으로 안드로이드 스케줄러를 사용할 수 있다. BroadcastReceiver 클래스를 상속받은 사용자 정의 클래스를 반복할 시간과 함께AlarmManager에 등록하면 설정한 시간에 맞추어 안드로이드 시스템에서 사용자 정의 클래스를 자식프로세스로서 반복적으로 호출하게 된다. 사용자 정의 클래스의 작업이 끝났다면 스케줄러가 안드로이드를 다시 sleep모드로 돌입하게 하여 배터리 과다소모를 방지해준다. 이 때 TCP소켓을 사용한다면 비 연결형 통신과 비교하여 긴 latency가 발생하게 되므로 HeartBeat 간격이 짧아질수록 배터리 소모가 큰 폭으로 커지게 되는 단점이 있기 때문에 사용자 내부 클래스에서 UDP소켓통신을 사용하여 HeartBeat인터페이스를 구현한다.

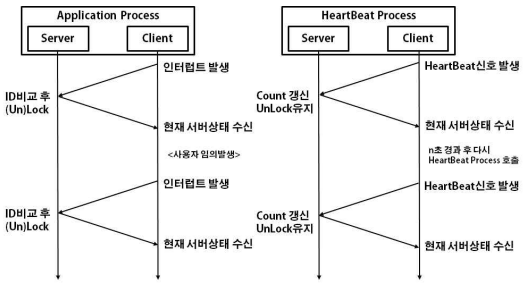

Fig. 3. Network Diagram

III. 보안 시스템 구현

1. System Overview

전체 시스템 구성도는 그림4와 같다. 그림 5는 최초등록 이후 보안 기능을 사용 가능한 시점에서의 시스템 흐름과 구조를 표현한 다이어그램으로서 USB 저장장치, Windows OS가 설치되어있는 PC, 안드로이드 스마트폰 세 디바이스의 실행흐름을 나타내고 있으며 USB인터럽트 발생과 안드로이드로부터의 인터럽트 발생 두 가지 시점으로 나뉜다.

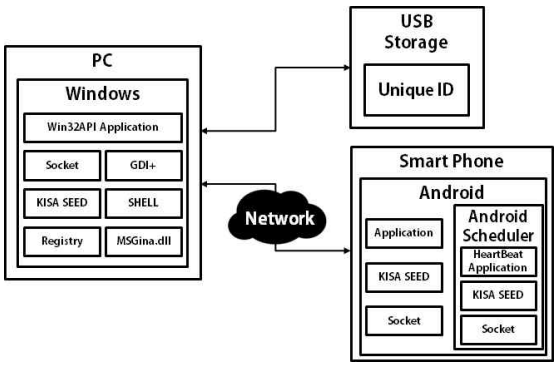

Fig. 4. System Diagram

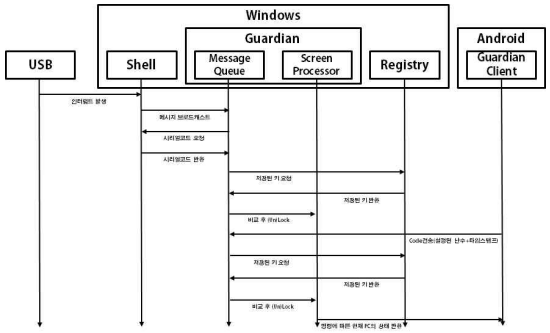

Fig. 5. Sequence Diagram

그림 5는 그림4의 과정을 모두 거친 후 최종적인 Lock 상태를 나타낸다. 이 상태에서는 Z Order 1순위 윈도우인 작업관리자, 작업표시줄 그리고 그 외의 모든 작업이 배제되며 Alt+F4와 같은 강제종료명령 핫키도 전역 후킹을 통해 메시지를 가로채어 명령을 무시한다.

2. Device Registration

본 보안 시스템을 사용하기 위해 USB와 스마트폰을 등록해야한다. 그림6은 그에 따른 등록윈도우를 표현하고 있다. 스마트폰 최초 등록 시 ID를 전송할 암호화키가 약속되어있지 않으므로 응용프로그램과 안드로이드 어플리케이션 내부에 약속되어있는 내부 비밀 키로 암호화 한 최초 암호화키를 서버와 클라이언트가 서로 나누어 가진다. 그 후 나누어가진 키로 ID를 타임스탬프와 혼합하여 SEED-CBC알고리즘으로 암호화 후 전송한다.

Fig. 6. Windows

3. Security Using a USB storage device

USB장치에 대한 변화가 발생했을 때 쉘은 인터럽트를 받아 본 응용프로그램으로 메시지를 전송한다. 본 응용프로그램은 쉘에게 요청하여 PC에 연결된 모든 USB저장장치의 고유ID를 가져와 레지스트리에 암호화하여 설정해둔 ID와 비교 후 PC를 보안 상태로 설정할지 판단한다. 고유 ID와 레지스트리의 ID가 일치하는 저장장치를 찾았다면 Lock화면을 해제하고, 일치하는 저장장치를 찾지 못하였다면 그림5의 Lock화면을 최상위 윈도우로 위치시켜 본 응용프로그램 이외의 모든 작업을 백 그라운드화 한다.

4. Security Using a SmartPhone

그림7의 UI는 그림3의 Application Process의 UI이다. 본 UI는 PC Lock상태와 PC UnLock상태를 UI에 표시하고 있다. 이처럼 사용자가 현재 PC의 상태를 알 수 있도록 클라이언트 어플리케이션은

로드 시 PC의 잠금 상태를 UI에 표시해야한다. 그러나 비 연결형 통신이므로 서버가 소켓연결을 감지할 수 없기 때문에 ScreenOn BroadCast가 감지된 직후 서버로 LINK메시지를 보내 현재 PC의 상태를 반환받는다. 그림2가 위 작업에 따른 패킷을 표현하고 있는데, 분홍색 패킷이 클라이언트가 보낸 패킷이며 명령 타입과 사용자 인증 ID를 담고 있다. 파란색 패킷이 클라이언트가 보낸 명령에 따른 반환을 나타내며 현재 PC의 잠금 상태를 담고 있다. 그림7의 중앙의 버튼을 누르게 되면 PC로 사용자 인증 ID를 전송한다. 서버가 사용자 인증 ID를 확인해 저장된 사용자 인증정보와 전송받은 사용자 인증 ID가 일치한다면 현재 PC 보안 상태의 반대상태로 전환하게 된다. Auto Locking을 설정하게 되면 일정 시간마다 서버로 HeartBeat 패킷을 보내 클라이언트가 서버 주변에 있음을 알리게 된다.(본 논문에서는 HeartBeat주기를 5초로 설정하였다.) 서버는 최초 HeartBeat패킷이 전송된 시점을 기준으로 AutoLock모드로 전환하게 되며, 10초 이상 HeartBeat가 전송되지 않을 경우 클라이언트가 AP를 벗어났다고 가정하여 서버PC를 그림5의 LockScreen으로 전환하게 된다.[7][8][9]

Fig. 7. Client Application

IV. 결 론

PC는 최초 로그인을 제외하면 현재 사용하고 있는 사용자가 누구인지를 판별할 수 없다. 또한 PC는 PC사용자의 주변 존재 유무를 알 수 없으므로 PC사용자가 PC를 공개된 상태로 자리이동을 했을 경우를 대비할 수 없다. 이러한 문제점을 개선할 방법으로 USB와 스마트폰이라는 장치를 개인 인증수단으로 사용한다면 PC를 획기적으로 보호할 수 있다고 판단한다. 따라서 USB의 고유ID와 스마트폰 최초설정시의 난수ID를 보안시스템이 소유하며 사용자를 인증하고 HeartBeat기술을 통해 PC관리자가 AP주변에 있음을 감지한다면 보다 완벽한 서비스를 제공해 사용자가 높은 수준의 물리적인 보호를 받고 있다고 인지할 수 있을 것 이라고 판단한다.

Reference

[1] KISA, "A study of secure alternative method of traditional alphanumeric password" Korea Internet & Security Agency, INHA University pp .15, Jul. 2009, https://www.kisa.or.kr/jsp/common/libraryDown.jsp?folder=013723

[2] KISA, "For block cipher SEED Source code utilizing manual" Korea Internet & Security Agency pp .5-17, Dec. 2013, http://seed.kisa.or.kr/iwt/ko/bbs/EgovReferenceDetail.do?bbsId=BBSMSTR_000000000002&nttId=77&pageIndex=1&searchCnd=&searchWrd=

[3] In-Kyong jeon, "Domestic usage and password Password Implementation Guide" Korea Internet & Security Agency pp .5-13, Nov. 2011

[4] Tue Christopersen, "Zero Interaction Multi-factor Authentication" Kongens Lyngby, pp. 77, 2010, http://etd.dtu.dk/thesis/276333/Tue_Christophersen.pdf

[5] MicroSoft MSDN, "GINA" MSDN,, 2010, https://msdn.microsoft.com/en-us/library/windows/desktop/aa375457(v=vs.85).aspx

[6] Sang-Hyeong Kim, "Windows Application Programming Interface Dominate-Vol2" Published by HANBIT Media, Inc Printed in korea, pp. 385-439, Oct. 2012

[7] seong-woo Yun, "TCP / IP SocketProgramming" Orenge Media, pp.147-168, Dec. 2012.

[8] Nakanishi Aoi, Uchimura Yuuji, Takahashi Ryouji "Android SDK Reference Book" kyohaksa publishing, pp. 829-830, 885-886, 860-861, 864-865, 988-989, Jan. 2015.

[9] Jae-Nam Woo, Gil-Sik Park, "Android Programming Using the Android Studio" hanbit Academy, pp. 194-266, 511-532, Jan. 2015.