중앙 집중형 컴퓨팅 시스템의 사용자 명령 인증 시스템 개발

김영호, 김성철, 원용관
전남대학교 컴퓨터공학과

e-mail:{melchi, sckim, ykwon}@grace.chonnam.ac.kr

Authentification of User Commands for Centralized Computing System

Y. H. Kim, S. C. Kim, Y. K. Won
Dept of. Computer Engineering, Chonnam University

요약

Computer Resource의 대용량화 및 인터넷 속도의 발달은 원거리에서 사용자가 인터넷을 통해 서버에 접속하여 컴퓨터를 사용하는 요구를 증가시켰다. 이는 Server-based 원격 컴퓨팅을 활용한 서비스의 연구 개발을 활성화 시켰으며, 최근들어서는 Thin Client를 기반으로하는 중앙 집중형 원격 데스크탑 시스템(Remote Desktop System)을 꾸준히 발달시켰다. 그러나 단순한 파일 데이터의 접근에 대한 보안만을 제공하는 현재의 보안 체계는 이러한 Server-based 컴퓨팅 환경에서 다수의 사용자에 대한 복잡한 형태의 정보 공유 및 미세한 수준의 접근 권한이 요구되는 경우에는 적합하지 않다. 본 논문에서는 ASP(Application Service Provider) 컴퓨팅 시스템과 같은 중앙 집중형 원격 데스크탑 시스템에 적용하여 하나의 응용 프로그램에 대해 사용자별 세부적인 하위명령 사용 제한을 수행하는 사용자 명령 인증 시스템을 제안한다.

1. 서론

컴퓨터와 인터넷의 발달은 Server Based Computing Model 또는 중앙집중식 서버(Centralized Server)라고 불리는 시스템의 발달을 촉진하였으며, 사용자로 하여금 GUI(Graphic User Interface)환경을 통하여 원거리에 존재하는 응용 프로그램의 수행을 보편화하였다[1][2]. 예를 들어, 소프트웨어 교육기관에서 많은 PC에 고가의 소프트웨어를 구매하여 설치 할 수 없기 때문에 Thin Client Solution[1]를 통한 응용 프로그램의 실습 교육이 이루어지고 있다.

그러나 Centralized Server의 관리자가 파일 데이터 접근권한 설정을 한다고 하여도, 파일의 양과 종류가 많아짐에 따라 파일 권한 설정만을 사용하여 시스템 사용자의 복잡한 사용권한을 제어하는 것은 한계가 있다. 예를 들어, 하나의 응용 프로그램에 대하여 "사용자 A는 '읽기' 및 '저장' 기능은 가능하나 '글자모양' 기능은 사용할 수 없다"라는 미세한 수준의 명령어 사용 제한은 불가능하다. 또한, Thin Client 기반의 윈도우 환경은 사용자의 ID와 암호만을 통해 인증을 하기 때문에 사용자 인증 이후에는 사용자 행동의 감시나, 윈도우 환경에서 어플리케이션의 메뉴, 툴바의 사용과 같은 사용자 명령을 실시간으로 감시 할 수 없다. 즉 윈도우 환경에서는 사용자의 명령은 메시지에 의해 시스템에 전달되며 이를 통해 시스템은 어플리케이션의 명령을 동작시키게 된다. 따라서 여러 가지 컨트롤들로 구성되어 있는 윈도우의 경우 파일 데이터 접근 권한 뿐만 아니라 메시지 모니터링과 메시지에 의한 명령어 접근 권한의 설정이 더욱 적절하다.

본 논문에서는 이러한 문제점을 해결하기 위해 Centralized Server환경으로 마이크로소프트의 터미

널 서비스(Terminal Service)[2][3]를 사용하고 로그인 과정에서 사용자별 서비스의 설정(하위 명령어에 대한 사용 권한 설정)과 서버 측의 응용 프로그램 사용에 대한 실시간 메시지 모니터링 및 제어를 통해, Thin Client 기반의 컴퓨팅 환경에 대하여 사용자별 응용 프로그램의 하위 명령어 사용에 대한 권한 인증시스템을 제안한다.

2. 터미널 서비스의 고찰

터미널 서버는 서버 자체, 사용자 인터페이스 전송 프로토콜, 클라이언트 등의 세 부분으로 구성되어 있다[4]. 클라이언트는 사용자 인터페이스 전송 프로토콜로 연결하여 다양한 사용자 인터페이스 지원한다. 여기에 Citrix ICA 프로토콜[2][5]을 사용한다면 비윈도우 기반 클라이언트도 터미널 서버에 연결 할 수 있다.

터미널 서버의 실행과정은 운영체제가 실행된 후 터미널 서비스가 실행되며, 3389번 포트를 통하여 클라이언트의 연결 시도를 기다리고, 세션관리자(Session Manager)가 Session 0을 생성해 서로 다른 사용자 로그온 하였을 때 인터랙티브한 사용자의 프로세서를 생성하게 된다[4].

터미널 서버는 RDP(Remote Desktop Protocol)을 사용하여 다중 채널 컨퍼런싱 프로토콜인 T.128[6] 에 기반을 두고 있으며, 윈도우 데스크탑을 제공하는 클라이언트는 Win32 윈도우 응용 프로그램 형태로 제공된다. 그리고 이런 클라이언트를 운용하는데 필요한 메모리의 사용이 작아 Thin Client라고 부른다[5].

클라이언트는 그래픽과 텍스트를 포함한 모든 원격 데스크탑 출력을 다루며 키보드, 마우스나 로컬 디바이스에서 입출력을 받는다. 이때 입력받은 데이터는 Win32 메시지 형식의 데이터로 클라이언트의 캐시 기능에 따라 서버에 최소화된 데이터를 전송한다. 캐시기능이란 서버에서 실행되고 있는 이미지를 비롯하여, 마우스 키보드들의 데이터를 모두 전송하야 하지만 클라이언트에 똑같은 기능의 캐시를 두어 화면 구성의 최소의 데이터만을 전송할 뿐 실제로는 Thin Client에서 구성한다.

클라이언트의 접속으로 서버에서는 세션 관리자를 통해 미리 생성된 Session 0으로부터 사용자별로 Session ID를 생성하고 사용자의 입출력 메시지를 각 Session별로 클라이언트의 메시지에 따라 처리하게 된다.

클라이언트와 통신은 40비트의 키를 가진 RC4를 통해 암호화되며[4], 윈도우상의 보안 기술을 통해 각 사용자가 접속했을 때, 파일 데이터 접근 권한 및 사용 권한 보안 설정을 적용하게 된다.

그러나 이러한 보안 설정은 윈도우의 많은 파일과 윈도우 Message형태의 명령 처리에 따라 각 사용자별로 완벽한 보안을 하기에는 부족한 하다. 또한 한 프로그램에서 DLL(Dynamic Link Library)과 여러 컴포넌트를 사용하여 작업을 하는 윈도우의 경우 이러한 현상이 더욱 두드러지게 나타난다. 이러한 문제를 해결하는 위하여 사용자가 서버에 접속했을 때 메시지를 관리하는 윈도우 서비스를 Session 내에 생성한다. 이 서비스에서는 실시간으로 처리되는 메시지 큐(Message queue)를 모니터링하고[7], 메시지 훅킹(Message Hooking)[8]을 통해 사용자 감시 및 인증 문제를 극복하고자 한다.

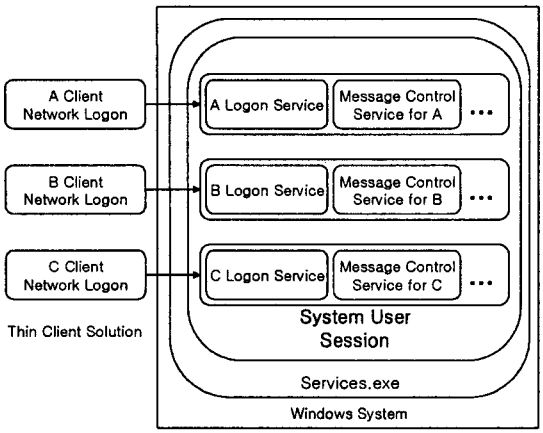
3. 세션을 위한 서비스 설계

사용자가 클라이언트프로그램을 통해 터미널 서버에 접속하면 서버는 Session Manager를 통해 각 사용자의 세션 및 서비스를 실행하게 된다. 만약 메시지 훅킹을 서비스가 아닌 어플리케이션에서 구성하게 된다면, 사용자가 이 어플리케이션을 종료 할 가능성이 있다.

따라서 본 논문에서는 관리자 이외의 사용자가 메시지 훅킹 과정의 종료를 방지하기 위해 일반 사용자가 제어할 수 없는 윈도우 서비스[9]를 구성하였다. 윈도우 서비스는 서비스 제어 관리자(SCM: Service Control Manager)를 통하여 관리되며, 서비스는 유형과 서비스 시작 방법에 따라 여러 가지 속성을갖는다.

본 논문은 서비스의 전제 조건으로 클라이언트가 접속하면 세션 생성과 동시에 모니터링 서비스가 실행되어야 한다. 따라서 사용자의 행동을 모니터링하기 위하여, 사용자 한 명에 서비스 하나로 구성된 SERVICE_WIN32_OWN_PROCESS 서비스 유형과 사용자별 세션이 시작되는 시점에 서비스를 시작하는 SERVICE_SYSTEM_START라는 시작방법을 선택하였다[9].

[그림 1]은 서버를 사용하는 사용자가 접속하였을 경우 서비스 형식으로 실행함으로써 사용자의 명령 인증을 향상시켜 각 사용자별로 서비스를 실행하기 때문에 다른 사용자의 메시지 명령에 영향을 받지 않고 메시지 훅킹을 할 수 있다[10].

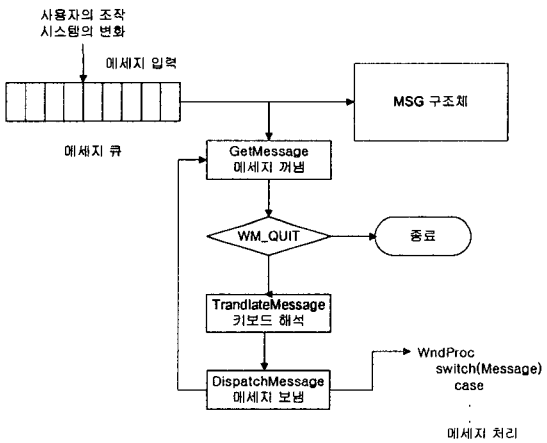


[그림 1] 사용자 메시지 제어 서비스

4. 윈도우 메시지 제어 및 훅킹

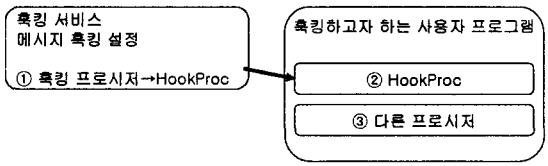
윈도우의 명령은 Unix처럼 Shell을 통한 응용프로그램 실행 비율보다 윈도우상의 마우스를 통해서 실행되는 경우가 많다. 또한 윈도우의 명령은 메뉴 또는 툴바 등의 Common Control Interface를 통해 명령을 메시지 큐에 추가함으로써 실행된다.

메시지 큐의 동작은 어플리케이션이 실행되고, [그림 2]처럼 사용자의 명령에 따른 입출력 메시지를 GetMessage함수를 통해 무한 루프를 돌며 MSG Struct형식의 메시지 큐에 저장한다[11]. 메시지 큐에 저장된 메시지는 저장된 순서에 따라 TranslateMessage와 DispatchMessage함수를 통해 전달하고 처리한다. 이때 각 메시지는 해당 윈도우 프로시저(WndProc)에서 처리하게 된다.



[그림 2] 메시지 처리 함수

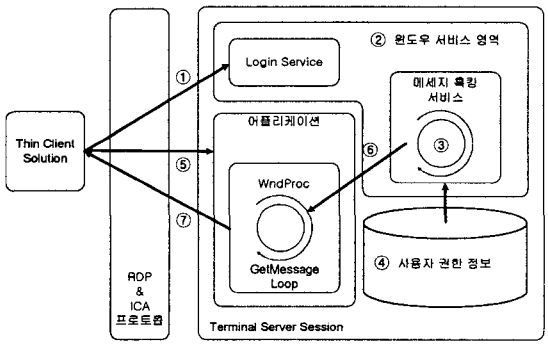
[그림 2]를 통해 애플리케이션 안에서 필요한 메시지를 받고 그에 해당하는 함수를 만드는 것에 대해 설명하였다. 그러나 메시지 정보를 얻기 위해서는 각 사용자의 메시지를 얻어야한다. 이런 과정을 메시지 훅킹이라고 한다. 메시지 훅킹은 시스템 및 어플리케이션 사이 또는 사용자의 입·출력 메시지 정보를 가로채고, 이를 수정, 삭제 할 수 있다. [그림 3]은 메시지 후킹을 간단히 나타낸것이다.



[그림 3] 메시지 훅킹의 방법

이러한 일련의 과정을 종합해 볼때 터미널 서비스를 통해 사용자가 접속했을때 사용자의 모든 메시지를 감시한다. 서비스 초기화에 사용자에게 허용되지 않는 Message ID를 정의하여, 사용자의 메시지가 발생하게 되면, 이때 메시지를 삭제하여 사용자가 해당 Message ID를 실행하지 않도록 한다.

5. 구현 및 실행



[그림 4] Hook을 위한 전체 구성도

이 처럼 Client의 요청에 연결이 생성되고 나면, [그림 4]는 이제까지 설명한 내용을 종합하여 흐름도를 구성한 것이다. 이 구성도는 다음의 순서로 작동을 하게 된다.

①을 통해 RDP 또는 ICA가 설정된 시스템에 사용자가 로그온하게 되고, ②에서 각 세션별로 윈도우 서비스를 시작한다. 이 서비스 관리자에서 메시지 훅킹 서비스 ③을 시작하고, ④의 사용자 권한

정보를 로드 및 후킹을 초기화한다. ⑤는 사용자에 의해 어플리케이션 실행을 나타내고, ⑥ 어플리케이션의 메시지를 서비스가 훅킹에 들어간다. 그리고 ⑦은 각 사용자에게 터미널 서비스의 캐시 기능을 사용하여 전달한다. 사용자의 로그인이 시작되면 위의 과정을 통해 사용자의 권한을 설정한다.

6. 결론

T.128 프로토콜을 통한 Terminal Server과 Thin Client Solution은 원거리의 사용자에게 윈도우 환경을 그대로 사용할 수 있게한다. 이러한 중앙집중형 컴퓨팅 시스템은 암호화된 프로토콜과 인증된 사용자에게 비주얼한 화면을 보여준다. 윈도우 시스템은 명령을 처리하는 방식이 메시지 방식으로 기존의 파일에 대한 데이터 액세스 제어만을 통해 사용자를 실시간으로 모니터링 하거나 사용자 명령에 대한 제어 권한을 설정하는데 어려움이 따른다.

본 논문에서는 중앙 집중형 원격 데스크탑 시스템에 적용하여, 중앙 서버에 위치하는 하나의 응용 프로그램에 대해 사용자별 세부적인 하위명령 사용 제한을 수행하는 사용자 명령 인증 시스템을 제안하였다. 제안한 인증 시스템은 각 사용자별로 응용 프로그램의 세부적인 하위 명령어에 대한 보안을 설정하고, 윈도우 환경에서 사용자의 명령을 처리하는 메시지를 훅킹하여 사용자의 행동을 실시간으로 감시하며, 매 명령어 마다 사용자의 보안 규칙과의 적합성을 판단함으로써 중앙 집중 서버의 응용 프로그램에 대한 각 사용자별 명령 인증을 수행한다.

제안한 사용자별 명령 인증 시스템은 관리자의 사용자별 보안 정책 설정 만으로 다수가 공동으로 사용하는 기업의 특정 업무 시스템에 대하여, 별도의 프로그램 변경 없이 "사용자 A는 '읽기' 및 '저장'은 수행 가능하고 '문단모양 변경'은 수행 불가능"하도록 하는 정책 기반의 응용 프로그램 사용 보안 (Policy-based Application Program Security) 시스템으로 발전하여야 한다.

참고문헌

[1] Borko Furht, "An Innovative Internet Architecture for Application Service Providers", Florida Atlantic University, Boca Raton, Florida.

[2] Todd W. Mathers. "Windows NT/2000 Thin Client Solutions:Implementing Terminal Services and Citrix MetaFrame". Macmillan Technical Publishing U.S.A.

[3] Windows 2000 Terminal Services Technologies "http://www.microsoft.com/windows2000/technologies/terminal/default.asp".

[4] Frank Kim. "Run Your Applications on a Variety of Desktop Platforms with Terminal Server". Microsoft System Journal 1999.1.

[5] "Server-Based Computing", Citrix Systems, While Paper, http://www.citrix.com, 1999.

[6] Microsoft, PictureTel, Polycom, "ITU-T recommendation T.128 SHARE - Application Sharing" Geneva, March 17th - 27th 1997.

[7] Microsoft Platform SDK Documentation: Message Queuing, "Messages and Message Queues", http://msdn.microsoft.com.

[8] 이상엽, "Visual C++ Bible" 영진출판사.

[9] Platform SDK: Windows User Interface "Windows Management Hook", http://msdn.microsoft.com/library/en-us/winui/hooks_9rg3.asp?frame=true.

[10] Keith Brown, "Programming Windows Security:develop mentor series" Addison Weskey.

[11] 김상형 " API정복:Windows Application Programming Interface", 가남사.