이동코드를 위한 통합 인증 시스템

배 성 훈[†] · 이 수 현[††]

요 약

자바, 자바스크립트, 액티브X, 스크립트 코드와 같은 실행가능 컨텐츠 또는 이동코드는 사용자가 인식 없이 서버에서 클라이언트로 전송되어 실행된다. 클라이언트로 전달되는 이동코드에 악의적인 내용이 포함되어 있다면 클라이언트 시스템의 정보를 외부로 유출, 파괴, 변경될 수 있다. 이동코드의 인터프리터들은 시스템을 보호하면서 효율적인 작업 수행을 도울 수 있는 보안모델을 가지고 있거나 이를 위한 도구들이 있는데, 이들은 서로 다른 기술로 개발되었기 때문에 공통으로 사용 가능한 하나의 보안모델을 가지고 있지 않아 관리의 어려움이 있다. 본 논문에서는 여러 종류의 이동코드로부터 사용자의 로컬 시스템을 보호할 수 있는 시스템 설계를 제시하고, 이를 바탕으로 통합인증시스템을 구현하였다. 통합인증시스템은 각 이동코드 별로 시스템 접근에 관한 접근제어목록을 기초로 운영되며, 이동코드의 사용 인증, 이동코드의 시스템 접근 인증, 웹 인터페이스를 이용한 접근제어목록 관리로 구성되어 있다.

An Integrated Authentication System for Mobile Codes

Sung-Hoon Bae[†] · Su-Hyun Lee[††]

ABSTRACT

Mobile codes such as Java, Java-Script, ActiveX, and Script code are loaded into a client system first and then run without any notice to the client user. Executing code by this mechanism may cause various security problems such as flowing out system information, deleting or modifying files, and exhausting system resources. In this paper we propose an integrated authentication system to establish the uniform security countermeasure on various mobile codes. The system helps to solve to problems mentioned above. An integrated authentication system allows to load into an interpreter using ACL (Access Control List) which sets up an access authority to the executable contents and communicates with an interpreter using client/server model.

키워드 : 이동코드(Mobile Code), 실행가능 컨텐츠(Executable Content), 보안(Security), 인증(Authentication), 공개키(Public Key), DTD(Document Type Definition)

1. 서 론

웹 페이지는 최근 점차 멀티미디어적인 요소가 추가된 동적인 화면으로 발전하고 있다. 동적 화면을 위한 대표적인 관련 기술들에는 자바, 자바스크립트, ActiveX, CGI(Common Gateway Interface), Server Side Script, Flash 등이 있다. 이들 중에 자바, 자바스크립트, ActiveX와 같이 클라이언트로 이동된 후 일정한 작업을 수행하는 것을 실행가능 컨텐츠(executable content) 혹은 이동코드(mobile code)라고 부른다[1]. 이러한 이동코드는 일반적으로 특정 하드웨어에 의존하지 않고, 동적 화면 구성을 위한 공통적인 작업을 가능하게 하는 편리함과 기능적인 장점을 가지고 있어 많이 사용되고 있다[2]. 이러한 장점들은 이동코드를 수행하는 인터프리터의 성능에 의하여 최대화될 수 있다. 만약 클라이언트로 전달되는 이동코드에 악의적인 내용이 포함되어 있다면, 인터프리터에 존재 가능한 특정 버그들을 이용하여 클라이언트 시스템의 정보를 외부로 유출, 파괴, 변경하는 것을 쉽게 할 수 있다[3, 4]. 따라서 이동코드의 수행은 무엇보다 안전하고 믿을 수 있는 인터프리터를 사용하여야 한다.

각각의 이동코드의 인터프리터들은 시스템을 보호하면서 효율적인 작업 수행을 도울 수 있는 보안모델을 가지고 있거나 이를 위한 도구들이 있다[5, 6]. 또한 사용자가 보안정책을 수립하여 사용자 시스템으로 전달되는 이동코드를 어떠한 방식으로 수행할 것인지를 결정하기도 한다. 하지만 각각의 이동코드는 서로 다른 기술로 개발되었기 때문에 공통으로 사용 가능한 하나의 보안모델을 가지고 있지 않고, 여러 시스템에 같은 보안정책을 적용하기 위해서는 각각의 시스템마다 똑같은 설정을 반복하여야 하는 관리의 어려움이 있다. 또한 다양한 이동코드가 서로 비슷한 방식의 보안모델을 가지고 있으며, 기존의 인터프리터들은 서로

※ 이 논문은 2000년도 창원대학교 연구비에 의하여 연구되었음.
† 정 회 원 : (주)애드컴인포메이션
†† 종신회원 : 창원대학교 컴퓨터공학과 교수
논문접수 : 2001년 8월 4일, 심사완료 : 2001년 11월 5일

다른 컴퓨터에서 같은 코드로 수행 가능하지만 외부 코드를 직접 로드하여 실행하는 경우를 위한 보안모델은 가지고 있지 않다.

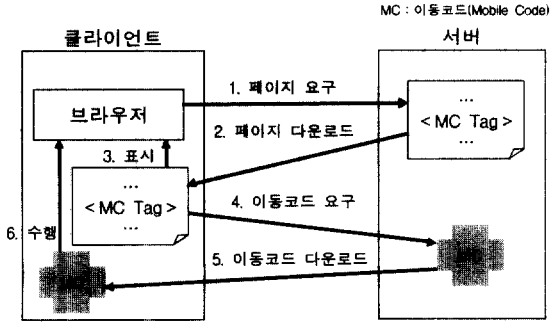
본 논문은 이동코드를 사용함에 있어서 사용자마다 자신에게 맞는 보안정책을 수립하고, 통합 인증 시스템을 통해 자신이 설정한 보안정책으로 클라이언트로 이동된 이동코드로부터 자신의 컴퓨터를 안전하게 보호하는데 목적을 두었다. 이를 위해서 이동코드가 믿을 수 있는 서비스 제공자에 의해 만들어졌다는 확인 작업은 전자서명과 코드위치를 이용하고, 인터프리터가 코드를 수행하는 동안에는 서버와의 통신을 통해 클라이언트 시스템 자원 접근 유무를 검사하도록 했다.

본 논문의 전체적인 구성은 다음과 같다. 먼저 서론에 이어, 2장에서는 이동코드와 보안모델 대하여 살펴보고, 3장에서는 본 논문에서 제안한 통합 인증 시스템의 설계에 대해서 기술한다. 4장은 3장의 설계를 기반으로 구현에 대해 기술한다. 마지막으로 5장에서 결론을 맺는다.

2. 관련 연구

2.1 이동코드

일반적으로 하나의 프로그램을 사용하기 위해서는 프로그램의 설치작업이 필요하지만, 이동코드는 사용자의 요청 시점 혹은 사용자는 인식하지 못하지만 서버에서 클라이언트로 이동된 후 수행된다. 이는 서버와 클라이언트 사이의 상호 작용성을 높이는 효과가 있다. 이러한 방법의 대표적인 예로는 자바, 자바 스크립트, ActiveX가 있다. (그림 1)은 이러한 이동코드의 일반적인 동작 방식이다.



(그림 1) 이동코드 동작 구조

이동코드를 이용하기 위해서 클라이언트에서 웹 서버에게 해당 페이지를 요청하면 웹 서버에서 해당 페이지를 클라이언트 전송한다. 전송된 웹 페이지는 웹 브라우저를 통해 그 내용이 표시되고, 웹 페이지 내에서 이동코드 처리 태그를 만나면 웹 브라우저는 웹 서버상의 이동코드를 요청하고 이동코드는 클라이언트 시스템으로 전송되어 클라이언트측의 인터프리터에 의해 해석되어 정해진 기능을 수행한다.

이동코드는 클라이언트와 서버 사이의 높은 상호작용이란 장점을 가지고 왔지만 외부의 알 수 없는 프로그램이 사용자 시스템에서 수행되기 때문에 보안과 관련된 여러 문제점이 존재한다. 보안과 관련한 몇 가지 사례를 [7], [8], [9] 등에서 살펴볼 수 있다. 문제 발생의 원인은 외부의 코드가 유해한지 그렇지 않은지를 판단할 명확한 기준이 없다는데 있다. 따라서 이동코드를 지원하는 시스템에 따라 외부 코드의 유해성 판단의 기준이 달라지게 되고 다양한 보안모델이 등장하게 되었다.

2.2 보안모델

이동코드의 보안 문제를 해결하기 위해 사용되는 일반적인 방법은 Sandbox와 전자서명 기법이다. Sandbox는 특정한 영역에 몇 개의 연산들만 실행하게 권한을 제한하는 방법을 통해 로컬 시스템의 프로그램과 외부 프로그램을 구별하였다. 전자서명을 이용한 방법은 이동코드가 특정인에 의해 작성됨을 전자서명을 이용해 확인 후 수행하는 방법이다. 이외에도 방화벽을 이용하여 클라이언트가 속한 도메인으로 이동코드가 들어오는 지점에서 코드를 수행할 것인지 여부를 결정하거나 Playground라는 공간에서 이동코드를 수행되도록 하고 I/O 결과만 사용자에게 보여지게 하는 방법이 있다[10].

기존에 많이 사용되는 스크립트들은 보다 안전한 인터프리터를 작성을 통해 코드이동을 통한 많은 혜택을 얻을 수 있다. TCL(Tool Command Language)은 인터프리터에 Sandbox와 비슷한 개념의 Padded Cell 방법으로 명령어 앨리어스 혹은 숨김을 통해 Safe-TCL로 발전하였다[11]. 비슷한 방법으로 웹 브라우저는 스크립트를 위한 보다 안전한 인터프리터를 작성하고, Python은 Safe-Python으로 발전하였다.

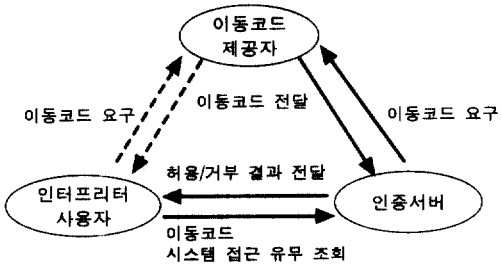
Vigna는 이동코드에 로그를 기록하여 이동코드의 유해여부 판별 할 수 있도록 하였다[12]. 하지만 이동코드를 처음 실행할 경우에는 적용될 수 없는 단점이 있다. Rubin 등은 하나의 시스템으로 이동코드를 위한 보안모델을 제안하였다[13]. 이것은 전달받은 이동코드를 인증하며, 이동코드가 로컬 시스템의 자원을 상용하기 위해서는 사용허가가 있어야 한다. 하지만 사용자가 다른 클라이언트를 사용할 경우 이전의 보안정책을 사용할 수 있는 단점이 있다.

3. 통합 인증 시스템

3.1 시스템의 설계

이동코드를 사용자가 이용하는 일반적인 경우는 (그림 1)과 같이 사용자가 직접 이동코드를 요청하여 다운로드 된 이동코드를 수행하는 것이다. 이러한 경우 이동코드 동작과 관련된 정보를 사용자는 가지고 있지 않으므로 이동코드

시스템에서 어떠한 작업을 수행하는지 알 수 없다. 이러한 문제 해결을 위해 본 논문은 이동코드로부터 로컬 시스템을 보호하기 위한 통합 인증 시스템을 제안한다. 기본적인 시스템의 동작도는 (그림 2)와 같다.



(그림 2) 통합 인증 시스템 기본 동작도

(그림 2)에서 점선의 경우는 기존의 이동코드 이용 방법을 나타낸 것으로 사용자가 이동코드 제공자에게 이동코드를 요청하여 다운로드 된 이동코드를 인터프리터를 통해 수행한다. 본 시스템에서는 인터프리터와 이동코드 사이에 인증서버를 두어서 사용자가 이동코드에 직접 접근하지 않고 인증서버를 이동코드 수행여부 혹은 이동코드 수행도중 로컬 시스템 자원 접근 여부를 알아보도록 하였다. 이를 위해 다음과 같은 기능을 설계하였다.

① 이동코드의 로컬 시스템 접근을 위한 접근제어리스트 (ACL ; Access Control List)의 정의
② 인터프리터와 시스템 간 통신을 위한 프로토콜 정의
③ 공개키 목록에 사용하기 위한 공개키 인증방법

첫 번째 항목과 세 번째 항목은 사용자가 인터프리터를 통해 이동코드를 사용할 때 이동코드와 관련된 접근권한을 판별하기 위한 기능으로 이동코드의 인터프리터에서의 수행여부와 사용자 시스템 접근 권한 여부를 판별하기 위한 기능이다. 두 번째 항목은 인터프리터가 ACL에 접근하기 위한 방법을 제공하기 위한 기능으로 인터프리터와 인증서버는 클라이언트/서버 모델을 이용하게 된다.

3.2 접근제어

인터프리터 혹은 웹 브라우저는 사용자가 미리 설정해둔 접근권한에 따라서 이동코드의 실행을 결정하기 때문에 통합 인증 시스템에서 이동코드의 처리 정보인 접근제어리스트(ACL)가 중요한 부분이다.

일반적으로 개인이 직접 작성하지 않은 이동코드는 그 작업과정에서 어떠한 일들을 수행하는지 정확히 알 수 없기 때문에 이동코드가 사용자 시스템의 자원에 접근하는 것을 막아야 한다. 하지만 때때로 이동코드를 수행함에 있어서 꼭 필요로 하는 로컬 시스템으로의 접근 작업이 있을 수 있다. 이런 작업에는 파일 읽기/쓰기, 소켓 열기 등과 같은 것이 있다. 본 시스템에서는 ACL에서 이러한 접근을 사용자가 지정한 내용에 한하여 수행하도록 설정한다.

제안된 통합인증 시스템의 ACL은 XML(eXtensible Markup Language) 형태로 이동코드의 인증 관련 정보를 저장한다. 통합 인증 시스템에서 사용할 ACL은 몇 가지 요소들의 일정한 나열을 통해서 구성된다. <표 1>는 ACL을 위해 정의된 DTD이다.

〈표 1〉 ACL DTD

```
<!ELEMENT ACL                (GRANT*) >
<!ELEMENT GRANT              (CONTENT*) >
<!ELEMENT CONTENT            (ALL|(AUDIO|FILE|ENVIRONMENT|
                                SOCKET)*) >
<!ELEMENT ALL                EMPTY >
<!ELEMENT AUDIO              EMPTY >
<!ELEMENT FILE               EMPTY >
<!ELEMENT ENVIRONMENT        EMPTY >
<!ELEMENT SOCKET             EMPTY >
<!ATTLIST GRANT              access      (private|public)  "private" >
<!ATTLISTCONTENT             name        CDATA             #REQUIRED
                             codebase    CDATA             #IMPLIED
                             signed      CDATA             #IMPLIED >
<!ATTLIST AUDIO              play        (enable|disable)  "disable"
                             record      (enable|disable)  "disable" >
<!ATTLIST FILE               path        CDATA             #REQUIRED
                             read        (enable|disable)  "disable"
                             write       (enable|disable)  "disable"
                             delete      (enable|disable)  "disable"
                             execute     (enable|disable)  "disable" >
<!ATTLIST ENVIRONMENT        read        (enable|disable)  "disable"
                             write       (enable|disable)  "disable"
                             delete      (enable|disable)  "disable" >
<!ATTLIST SOCKET             accept      (enable|disable)  "disable"
                             connect     (enable|disable)  "disable"
                             listen      (enable|disable)  "disable"
                             resolve     (enable|disable)  "disable" >
```

GRANT 요소는 ACL의 여러 권한들이 작성자뿐만 아니라 다른 사용자가 ACL을 참조하기 위해 접근할 수 있도록 access 속성(attribute)을 가진다. access 속성은 ACL 작성자만 사용 가능한 private와 다른 사용자에게 공개하는 public 속성값이 있으며, access가 명시적으로 지정되지 않으면 private로 처리한다. GRANT는 하나 이상의 CONTENT 속성들을 가질 수 있다. CONTENT 요소는 이동코드와 관련된 대부분의 정보들로 구성된다. CONTENT는 name, codebase, signed 속성을 가지며 이들의 적절한 조합을 통해서 이동코드가 로컬 시스템에서 수행되도록 해 준다.

로컬 시스템으로의 접근은 ALL, AUDIO, FILE, ENVIRONMENT, SOCKET 들로 표현된다. ALL 요소는 이동코드가 로컬시스템의 프로그램들과 같이 사용자 시스템의 모든 자원을 사용 가능하게 할 경우에 사용된다. AUDIO는 소리, FILE 파일/디렉토리, ENVIRONMENT는 인터프리터의 환경, SOCKET는 소켓 사용에 접근하기 위한 권한을 나타낸다. 각 요소가 가지고 있는 모든 속성값은 기본적으로 'disable' 값을 가지고 있으며 권한을 주고자 할 경우에는 'enable' 값을 주어야 한다. <표 2>는 DTD를 이용한 ACL 예이다.

〈표 2〉 ACL의 예

```
<?xml version = "1.0"?>
<!DOCTYPE ACL SYSTEM "acl.dtd">
<ACL>
        <GRANT access = "private">
                <CONTENT name = "hello.class"
                        codebase = "http://pl.changwon.ac.kr"
                                signed = "test">
                </CONTENT>
        </GRANT>

        <GRANT access = "public">
        </GRANT>
</ACL>
```

이 예제는 하나의 이동코드를 인증하는 조건으로 codebase가 http : //pl.changwon.ac.kr/를 기반으로 하고 test에 의해서 전자서명이 된 hello.class를 인터프리터가 실행하는 것을 허락한다. 하지만 이 이동코드가 로컬 시스템의 자원에 접근하는 권한은 부여하지 않는다.

3.3 메시지 교환 프로토콜

통합 인증 시스템은 하나의 로컬 시스템에서만 운용할 수도 있지만, 동일한 사용자가 서로 다른 시스템에서 자신이 미리 작성해 둔 ACL을 이용하여 재작성 없이 사용하기 위하여 인터프리터와 인증 서버 사이의 주고 받는 메시지 규칙이 필요하다. 본 논문에서는 클라이언트와 서버 사이의 메시지 송수신에서 발생할 수 있는 보안 문제는 제외되었는데, 이는 SSL을 이용하여 서버와 클라이언트가 주고 받는 정보를 암호화하는 방법을 통해 해결될 수 있다.

메시지는 클라이언트에서 서버로의 REQUEST와 서버에서 클라이언트로의 RESPONSE로 이루어진다. REQUEST는 4가지의 서비스 요청 메시지를 가지고 있으며 각각은 인증 시스템 연결/종료를 위한 LOGIN/LOGOUT과 이동코드의 사용 허용 및 로컬 시스템 접근 제어를 위한 AUTH_CONTENT/AUTH_ACCESS가 있다. RESPONSE는 REQUEST에 대한 응답 메시지로 REQUEST가 서버에서 처리 결과와 경우에 따라 추가적으로 서버에서 클라이언트로 전달되는 메시지 정보를 담고 있다. 인증 시스템을 사용하기 위해서는 미리 등록되어 있는 계정(id)과 암호를 LOGIN 메시지에 포함해서 보내야 한다. 그리고 인증 시스템의 사용이 종료되면 LOGOUT 메시지를 보내어 인증 시스템과의 접속을 종료한다. AUTH_CONTENT는 사용자가 사용하게 될 이동코드가 로컬 시스템에 전송되어 인터프리터 혹은 웹 브라우저에서 실행 가능한지 확인하기 위한 메시지이다. 이 메시지는 클라이언트 쪽에서 수행을 원하는 이동코드의 위치를 URL 방식의 표현방법으로 정보를 주어야 한다. <표 3>은 서버와 클라이언트가 주고받는 기본적인 메시지 프로토콜을 보여준다.

〈표 3〉 메시지 교환 프로토콜

```
MESSAGE          = REQUEST | RESPONSE
REQUEST          = LOGIN | LOGOUT |
                   AUTH_CONTENT | AUTH_ACCESS
RESPONSE         = MESSAGE_STATE
LOGIN            = "LOGIN" USER_INFORMATION
LOGOUT           = "LOGOUT"
USER_INFORMATION = < 사용자 식별 ID > < 사용자 패스워드 >
AUTH_CONTENT     = "AUTHCONTENT" URL
URL              = < HTTP의 URL 표기법 >
```

AUTH_ACCESS는 실행중인 이동코드가 로컬 시스템의 자원을 사용하기 위한 접근 허용 여부를 알아 보기 위한 메시지이다. 이 메시지는 수행중인 이동코드를 나타내는 이름과 사용 허락을 받고자 하는 로컬 자원의 RESOURCE_REQUEST 정보들의 조합으로 구성되어 있다. MESSAGE_STATE는 클라이언트가 서버로 보낸 메시지 처리 결과를 나타낸다. 이 결과는 세 자리 숫자로 이루어져 있으며 <표 5>에 각 숫자의 의미가 정리되어 있다. 클라이언트는 이

〈표 4〉 시스템 자원 관련 메시지

```
AUTH_ACCESS = "AUTHACCESS" <Content name>
                RESOURCE_REQUEST
RESOURCE_REQUEST = AUDIO_TYPE | FILE_TYPE |
                   ENVIRONMENT_TYPE |
                   SOCKET_TYPE
AUDIO_TYPE = "AUDIO" ( "play" | "record" )
FILE_TYPE = "FILE" ( "read" | "write" | "delete" | "execute" )
            < 파일 경로 >
ENVIRONMENT_TYPE = "ENVIRONMENT" ( "read" | "write"
                   | "delete" )
SOCKET_TYPE = "SOCKET" ( "accept" | "connect" | "listen"
              | "resolve" )
MESSAGE_STATE = RET_MSG_CODE [ <size_length>
                <content_length> <content> ]
```

〈표 5〉 상태 코드 일람표

상태 코드	메시지	설 명
200	SUCCESS	요청이 성공적으로 이루어 졌을 때 전달된다.
201	SE_TRANS_CONTENT	AUTHCONTENT 메시지에 의해 Content의 인증이 성공적으로 이루어졌을 때 전달된다.
220	ALLOW_ACCESS	AUTHACCESS 메시지에 의해 Content가 요청한 시스템 접근 허용 허락할 때 전달된다.
400	FAILURE	요청이 성공적으로 실패하였을 때 전달된다.
401	CMD_ERROR	요청 메시지가 형식에 맞지 않을 때 전달된다.
402	CMD_NOT_FIND	잘못된 요청 메시지일 때 전달된다.
410	DENY_CONTENT	AUTHCONTENT 메시지에 의해 Content의 인증이 실패하였을 때 전달된다.
420	DENY_ACCESS	AUTHACCESS 메시지에 의해 Content가 요청한 시스템 접근 허용을 허락하지 때 전달된다.
500	NO_LOGIN	아직 시스템 LOGIN이 되지 않았을 때 전달된다.

숫자를 확인하여 서버로 보낸 메시지가 정확히 수행되었는지 또는 메시지가 왜 잘못되었는지 또는 서버에서 클라이언트로 보내는 다음 메시지가 어떠한 형태로 되어 있는지 등과 같은 정보를 얻을 수 있다. <표 4>는 시스템 자원 접근 요청과 관련된 메시지 프로토콜을 보여준다.

3.4 공개키 관리

공개키 인증 방식은 이동코드에 전자서명이 주어지는 경우 전자서명의 유효성을 확인하는 부분에 이용된다. 사용자는 다른 사람의 공개키가 자신이 알고 있는 그 사람의 것이라고 믿고 사용하기 때문에 매우 중요한 문서를 암호화해서 보내거나 공개키 생성자가 서명해 보내온 자료를 확인할 수 있다. 그러나 공개키가 정말 자신이 믿고 있는 그 사람에 의해 생성한 것인지를 판단하기는 매우 어렵다. 자신이 신뢰하는 사람들이 인정한 공개키라면 그 공개키의 생성자는 제 3자에 의해 인증되어지고 그 공개키를 이용해 메시지를 주고받을 수 있다. 공개키와 관련된 이러한 환경을 제공하기 위해 많은 사람들이 PKI(Public Key Infrastructure)에 관해서 연구를 하고 있다[14, 15].

본 논문에서는 공개키 관리에 있어서 PGP(Pretty Good Privacy)[16]의 개인의 신뢰에 근거한 상호인증 형식을 취하여 각 사용자가 ACL과 함께 사용할 공개키는 사용자 개인의 신뢰에 근거하도록 한다. 이러면 서버의 구성에 있어서 공개키의 인증과 관련된 부담을 줄여 주지만 사용자는 그만큼 자신이 공개키에 관해서 많은 점검을 필요로 하게 된다.

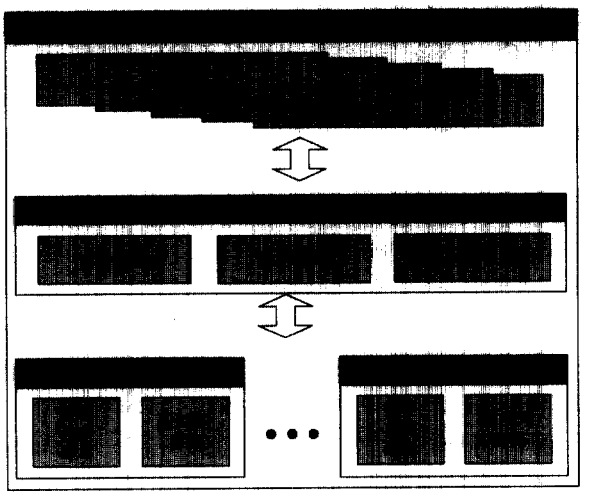
4. 구 현

3장에서 제안된 통합 인증 시스템을 위한 ACL과 메시지 교환 프로토콜을 이용하여 통합인증서버와 간단한 이동코드 해석기를 작성하였다. 본 논문에서 개발한 시스템의 구현 환경은 Intel 기반의 하드웨어 장비에 Linux 운영체제를 사용하였다. 서버 시스템 개발에는 Python, GCC, PGPsdk[17]를 이용하였으며 사용자 웹 인터페이스는 PHP를 이용하여 작성하였다.

4.1 서 버

서버 프로그램은 인터프리터 혹은 웹 브라우저와 네트워크를 통해 연결되어 이동코드가 수행되는 과정에서 이동코드의 클라이언트 시스템 자원 접근 여부판별 기능과 이동코드 인증 기능을 담당한다. 서버 프로그램의 전체 구성도는 (그림 3)과 같다.

통합 인증 시스템은 크게 세가지 서비스로 구성된다. 인증관리 모듈은 이동코드를 인터프리터 혹은 웹 브라우저에서의 실행여부를 결정하며, 접근관리 모듈은 이동코드의 실행과정에서 로컬 시스템의 자원 접근 여부를 결정하고, 인증리스트관리 모듈은 사용자의 ACL을 관리한다. 이러한 작



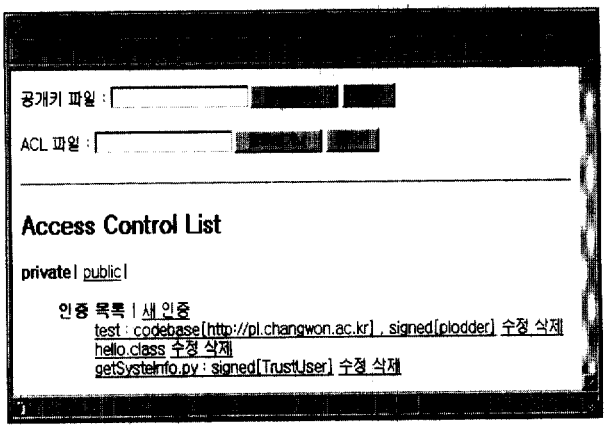
(그림 3) 통합 인증 서버 프로그램

업은 3장에서 소개한 ACL과 메시지 교환 프로토콜을 이용하여 이루어진다. 또 통합 인증 시스템은 여러 인터프리터(및 다중 사용자)와 통신을 하기 위한 부분은 스레드로 구성하였고, 각 사용자별로 적용될 ACL과 공개키 목록을 가지도록 하였다.

공개키 목록과 관련하여 인터프리터로부터 이동코드를 사용하기 위한 요청이 들어올 경우 서버는 ACL을 이용하여 사용자 로그인 인증 작업 후 사용자에 의해 인증된 이동코드일 경우 인터프리터를 대신하여 주어진 주소에서 이동코드를 가져온다. 만약 ACL에서 전자서명된 이동코드를 허락하면 인터프리터가 요청한 주소에 ".sig"를 붙여 이동코드를 가져온다. 그렇기 때문에 이동코드 제공자는 PGP를 이용하여 자신의 전자서명을 추가하여 ".sig"로 끝나는 또 하나의 이동코드를 만들어 두어야 한다.

4.2 사용자 웹 인터페이스

ACL 리스트를 사용자가 ACL DTD 형식에 맞게 수작업으로 편집하는 것과 사용자가 사용할 공개키 목록을 만드는 것은 쉽지만 시스템 관리나 프로그래밍에 익숙하지 않



(그림 4) 사용자 웹 인터페이스

은 일반 사용자들에게는 쉽지 않은 일이다. 이러한 이유 때문에 ACL 목록을 편집하고 공개키 목록을 관리해 주는 사용자 웹 인터페이스 프로그램을 작성하였다. (그림 4)는 사용자 웹 인터페이스의 메인 화면이다.

4.3 통합 인증 시스템을 사용한 인터프리터

이동코드를 수행하는 인터프리터는 이를 안전하게 수행하기 위해서 통합인증시스템과의 통신을 통해 이동코드 인증과 로컬 시스템 접근 권한 여부를 알아낸다. 인터프리터는 먼저 이동코드가 로컬에서 로드되는지 아니면 외부에서 로드되는지를 조사하고 만약 외부에서 로드되면 통합인증 시스템에 접속을 시도한다. 그리고 실행하려는 이동코드가 ACL에 정의된 형태를 통해 인증 절차를 거쳐 수행 과정에서 로컬시스템으로의 접근 권한이 있는 경우 서버를 통한 접근 가능 여부를 조사한다. 다음은 Python 소스로 작성된 인터프리터의 일부분이다.

```
...
# 이동코드를 로드한다.
if ( localcontent ) : # 로컬 시스템에 있는 경우
try :
f = fopen ( content, "r" )
            _content = f.read( )
            f.close ( )
      except :
            print "Content 로드를 실패하였습니다."
            sys.exit ( )
   else :      # 외부 시스템에 있는 경우
      try :
            err, _content = trust.getContent ( content )
            if(trust.isError ( err )) :
              print trust.getErrorString ( err )
              sys.exit ( )
      except :
            print "Content 로드를 실패하였습니다."
            sys.exit ( )
...
# 이동코드를 수행중 로컬 시스템에 접근이 필요한 경우
if ( not localcontent ) :
   # 외부 시스템으로부터 실행된 이동코드일 경우
   err = trust.isEnvAccess ( tsdk.FILE_READ, FILEPATH )

   if ( trust.isError ( err )) :
        print trust.getErrorString ( err )
sys.exit ( )
...
```

4.4 적용 예

이번 절에서는 서버 프로그램 인터프리터를 이용하여 Python으로 작성된 동일한 코드가 로컬 시스템에서 로드되어 수행되는 결과와 외부 시스템에서 로드되어 ACL을 이용한 접근 제어를 통해 수행되는 결과를 실험하여 실제 응용분야에서 유용하게 적용될 수 있음을 보였다. 작성된 인터프리터는 Python 코드를 수행하면서 처리 결과를 출력하도록 하였으며 실험에 사용된 Python 코드는 다음과 같다.

```
from socket import *
s = socket ( AF_INET, SOCK_STREAM )
s.connect (( "pl-serv.changwon.ac.kr", 80 ))
ret = s.send ( 'GET /~plodder/test.html\n\n' )
data = s.recv ( 1024 )
s.close ( )
f = open ( "/tmp/index.html", "w" )
f.write ( data )
f.close ( )
f = open ( "/tmp/index.html", "r" )
print f.read ( )
f.close ( )
```

위 소스코드는 네트워크에 연결된 다른 컴퓨터에 접속하여 특정 문서의 내용을 받은 후에 로컬 시스템의 특정 위치에 그 내용을 저장한 후 다시 이를 출력하는 일을 한다. 이 예제에서는 소켓과 파일에 대한 로컬 시스템 접근을 시도한다. 실험에서 점검되는 부분은 다음과 같다.

- 이동코드의 로드 성공 여부
- 외부 시스템 접속을 위한 connect 함수 호출 성공 여부
- 로컬 시스템에 파일 작성을 위한 write 함수 호출 성공 여부
- 로컬 시스템의 파일 내용을 읽기 위한 read 함수 호출 성공 여부

4.4.1 로컬 시스템의 이동코드 수행

4.3절에서 작성된 인터프리터는 로컬 시스템에 존재하는 코드에는 특별한 제약을 두지 않고 이동코드를 실행하기 때문에 소켓 열기, 파일 쓰기, 파일 일기와 같은 시스템 자원에 접근하는 것을 허용한다.

```
Content 위치 : 내부시스템
Content 로드 성공
< html >
< head >
    < title > 이동코드< /title >
< /head >
< body >
    이동코드에 대하여…
< /body >
< /html >
```

4.4.2 외부 시스템의 이동코드 수행

이동코드가 외부 시스템에 존재하는 경우는 인터프리터는 시스템 자원 접근에 대하여 제한을 한다. 다음은 외부의 이동코드에 대한 제한을 주는 ACL의 일부분이다.

```
< CONTENT name = "test.py"
          codebase = "http://pl.changwon.ac.kr/~plodder" >
    < SOCKET connect = "enable"/ >
< /CONTENT >
```

이 ACL에서 사용자는 위와 같이 이동코드의 이름과 위치

가 맞는 경우에만 소켓 열기를 허락하고 나머지 시스템 자원 접근을 제한한다. 아래의 인터프리터를 통해 수행된 출력 결과를 살펴보면, 파일 작성을 위한 write 부분에서 시스템 접근 거부가 발생하였다. 이는 위에서 허락한 소켓을 이용한 외부 접속은 허락되었으나 로컬 파일 시스템의 쓰기 작업이 거부되어 인터프리터가 종료되었음을 보여준다.

```
Content 위치 : 외부시스템
사용자 인증 성공
  Content 로드 성공
   --> (420) write : ACL에 의한 시스템 접근 거부
```

4.5 기존 시스템과의 비교

이동코드를 위한 기존 시스템들은 특정한 언어를 지원하고 그 언어에 맞는 보안모델을 가지고 있지만 타 언어에 보안모델 적용하지 못하는 단점을 가지고 있다. 이의 해결책으로 여러 이동코드를 동일한 보안모델을 적용하기 위한 방법들이 연구되고 있다. Vigna[12]는 이동코드를 처음 수행한 후 여기에 수행 로그를 붙이는 방법을 이용하였으며, Rubin[13]은 보안모델 사용을 위한 인터프리터 수정과 ACL을 통해 시스템 접근을 제한하려는 방법을 선택하였다. Rubin의 시스템과 자바가상기계[18]는 독자적인 형식의 ACL을 제공한다. 제안 시스템은 XML 형식을 이용하여 ACL을 제공하여 시스템 접근 보안모델이 변경될 경우 DTD 수정을 통해 간편히 할 수 있도록 하였다.

자바 가상기계의 경우는 ACL이 로컬 시스템의 파일 시스템에 존재하고 있으며, A. Rubin의 논문에서는 분산파일 시스템을 통한 ACL 접근을 제공한다. 제안 시스템은 클라이언트/서버 모델을 통해 인터프리터가 인증 서버를 통해 요청/응답 형태의 접근을 제공하였으며, 네트웍이 연결되어 있는 어느 곳에서든지 이용할 수 있도록 하였다. <표 6>에서 이동코드 수행 환경을 위한 시스템들을 비교하였다.

〈표 6〉 기존 시스템과의 비교

	제안시스템	Rubin[13]	Vigna[12]	자바가상기계
지원 이동 코드 수	다양함	다양함	다양함	자바 애플릿
ACL을 이용한 접근 제어	○	○	×	○
ACL 수정의 편의성	○	△	—	△
ACL 정보 위치	외부 시스템	외부 시스템	—	로컬 시스템
ACL 접근 방식	클라이언트 /서버 모델	분산 파일 시스템	—	로컬 파일 시스템

4.6 분 석

본 논문에서 제안되고 구현된 부분은 인터프리터가 서버와의 통신을 통해 외부의 이동코드가 로컬 시스템에서 수행할 때 사용자가 미리 작성해둔 ACL 규칙에 맞게 이동코드를 제어하여 내부 시스템을 보호하는 방법이다. 따라서 인증과 접근제어를 담당하는 서버와 인터프리터 사이의 통신 과정에서는 다음과 같은 문제가 발생할 수 있다. 이러한 문제점들은 네트워크를 통해 주고받는 메시지를 암호화하는 방법을 통해 해결될 수 있다.

(1) 서버와 인터프리터가 주고받는 메시지의 보안 위험성 : 주고 받는 메시지가 암호화된 것이 아니기 때문에 누군가 메시지를 가로채 메시지를 해석할 수 있다.
(2) 누군가에 의해 서버와 인터프리터 사이에서 잘못된 정보를 서버 혹은 인터프리터에게 전달 할 수 있다.

4.4절에서 적용한 예에서 같은 코드로 작성된 이동코드가 로드되는 위치와 ACL에 따라서 다른 결과를 가져오는 것을 보았다. 다음은 본 논문에서 제안한 방법에 대한 분석 결과이다.

(1) 다양한 이동코드를 하나의 시스템 보안모델로 제어 가능하므로 이동코드의 인터프리터는 자신만의 특성을 위해 최소한의 보안모델만 가지면 된다.
(2) 서로 다른 보안모델을 적용하는 인터프리터는 각각 다른 접근 제어 규칙을 가지고 있었으나 제안한 방법을 통해 하나의 접근 제어 규칙을 가질 수 있다.
(3) 인증/접근제어 부분을 담당하는 서버가 동일한 시스템에 있지 않기 때문에 사용자가 다른 곳으로 이동하더라도 이전에 작성되어 있는 접근 제어를 어느 곳에서든지 이용할 수 있다.
(4) 기존의 이동코드 수정 없이 인터프리터의 수정만을 통해 간단히 제안 시스템의 보안모델을 적용할 수 있다.

5. 결 론

본 논문에서는 이동코드로부터 사용자의 로컬 시스템을 보호할 수 있는 시스템 설계를 제시하고, 이를 바탕으로 통합인증시스템을 구현하였다. 통합인증시스템은 각 이동코드별로 시스템 접근에 관한 ACL 목록을 기초로 운영되며, 이동코드의 사용 인증, 이동코드의 시스템 접근 인증, 웹 인터페이스를 이용한 ACL 관리로 구성되어 있다.

본 연구의 결과는 앞으로 많이 등장할 이동코드로부터 시스템을 보호 하는 방법 및 이동코드의 인증에 관한 연구에 도움을 줄 것이다. 또한 이동코드는 컴퓨터 운영 환경을 벗어나 무선이동통신 분야로 그 활용 범위가 확대 될 것이다. 이런 경우 작은 기억 용량을 가진 장치에서 각각의 이동코드 별로 인터프리터를 갖추거나 관련 보안 기능을 모두 포함하기는 힘들 것이기에 연구에서 제안한 방법을 이용한 기존 기능 및 새로운 기능의 추가와 개발 시간의 단

축이라는 측면에서 그 활용이 기대된다.

본 논문에서 구현한 내용은 사용자의 로컬 시스템으로 전송되어 온 이동코드를 수행하는 과정에서 로컬 시스템을 보호하는데 중점을 둔 것이기 때문에 통합운영시스템과 로컬 시스템 사이의 통신 보호를 위한 부분은 제외되어 있다. 이와 관련해서는 SSL을 이용하면 그 해결책이 될 것으로 보이며, 실제로 이런 부분에 관련하여 보다 다양한 방법으로 이동코드와 관련된 주변 장비 및 컨텐츠의 보호에 관한 연구가 필요하다.

참 고 문 헌

[1] W3C, "On Mobile Code," (http://www.w3.org/MobileCode/).

[2] 이정호, "웹 페이지 Executable Contents 현황", 한국정보보호센터, (http://www.kisa.or.kr/technology/sub4/EC_9904.html), 1999.

[3] 이정호, "콘텐트의 변화에 따라 불법적 접근 증가", 한국데이터베이스진흥센터, 데이타베이스월드, 1999.

[4] "Executable Content and Information Security," (http://www.upenn.edu/computing/security-privacy/java/help.html).

[5] C. Kerer, "A flexible and extensible security framework for Java," (http://www.infosys.tuwien.ac.at/Teaching/Finished/MastersTheses/JSEF/thesis.html), 1999.

[6] S. Gritzalis, G. Aggelis, and D. Spinellis, "Architectures for secure portable executable content," Internet Research, 9(1) : pp.16-24, 1999.

[7] D. Dean and D. S. Wallach, "Security Flaws in the HotJava Web Browser," Technical Report pp.501-95, Department of Computer Science, Princeton University, November, 1995.

[8] V. Anupam and A. Mayer, "Security of Web Browser Scripting Languages: Vulnerabilities, Attacks, and Remedies," Proc. 7th USENIX Security Symposium, 1998.

[9] E. W. Felten, D. Balfanz, D. Dean, and D. S. Wallach, "Web Spoofing: An Internet Con Game," Proc. 20th National Information Systems Security Conference, (http://www.cs.princeton.edu/sip/pub/spoofing.pdf), 1997.

[10] D. Malkhi, M. K. Reiter, and A. D. Rubin, "Secure execution of java applets using a remote playground," Proc. 1998 IEEE Computer Society Symposium on Research in Security and Privacy, 1998.

[11] J. K. Ousterhout, J. Y. Levy, and B. B. Welch, "The Safe-TCL Security Model," Technical Report SMLI TR-97-60, Sun Microsystems, 1997.

[12] G. Vigna, "Cryptographic Traces for Mobile Agents," Lecture Notes in Computer Science, Vol.1419, 1988.

[13] T. Jeaeger, A. Prakash, and A. Rubin, "Building systems that flexibly control downloaded executable context," Proc. 6th USENIX Security Symposium, 1996.

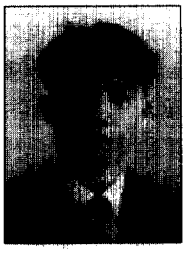
[14] W. T. Polk, D. F. Dodson, etc, "Public Key Infrastructure : From Theory to Implementation," (http://csrc.ncsl.nist.gov/pki/panel/overview.htm).

[15] "PKI의 정의", (http://ecommerce.infomail.co.kr/security3.htm).

[16] S. Garfinkel, "PGP : Pretty Good Privacy," O'Reilly, 1995.

[17] PGPsdk 홈페이지, (http://www.pgp.com/products/sdk/).

[18] B. Venners, "Inside the Java Virtual Machine," McGraw-Hill, 1998.



배 성 훈

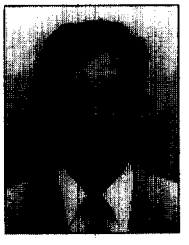
e-mail : plodder@hanmail.net

1999년 창원대학교 전자계산학과 졸업 (이학사)

2001년 창원대학교 대학원 전자계산학과 졸업 (이학석사)

2001년~현재 (주)애드컴인포메이션

관심분야 : Mobile Code, 보안, CDMA 이동통신, 리눅스



이 수 현

e-mail : suhyun@sarim.changwon.ac.kr

1987년 광운대학교 전자계산학과 졸업 (이학사)

1989년 한국과학기술원 전산학과 졸업 (공학석사)

1994년 한국과학기술원 전산학과 졸업 (공학박사)

1994년~1996년 한국전자통신연구소 선임연구원

1996년~현재 창원대학교 컴퓨터공학과 부교수

관심분야 : 프로그래밍언어, 컴파일러, WWW의 교육적 활용, Bioinformatics