웹 서버/클라이언트를 위한 보안 모듈 설계 및 구현

변용덕[0] 장승주
동의대학교 정보통신공학과
{ydbyun, sjjang}@hyomin.dongeui.ac.kr

Design and Implement of Security Module for Web Server and Client

Yong-Duk Byun[0] Seung-Ju Jang
Dept. of Information and Communication, Dongeui University

요 약

현재의 인터넷을 통한 웹 서버/클라이언트 환경에서 보안과 신뢰성 문제는 나날이 증가하고 있다. 기술적인 측면에서는 이러한 문제점을 개선하기 위하여 서버는 기존의 Apache 웹 서버에 라이브러리 형태의 보안모듈을 추가하였다. 보안 모듈의 기능은 클라이언트의 요청이 발생하면 웹 문서에 대한 RSA 암호화 기능과 메시지의 무결성 검사를 위한 SHA-1기능과 키 생성을 위한 랜덤 키 생성 기능을 포함한다. 클라이언트는 기존의 웹 브라우저에 Winsock2의 LSP 기능을 이용하여 보안 모듈을 체인의 형태로 삽입함으로써 보안 상의 문제점을 개선하고자 한다. 클라이언트의 보안 모듈의 기능은 서버로부터 받은 암호화된 메시지에 대한 RSA복호화 알고리즘과 메시지가 네트워크를 통해 전송되는 도중 변경되지 않았음을 증명하기 위한 SHA-1알고리즘을 포함한다. 그리고 사용자 편의성 측면에서 보안을 위한 새로운 소프트웨어의 설치와 기존의 프로그램 변경 없이 모듈을 추가, 삭제함으로써 사용자의 편리성을 추구 하였다.

1. 서 론

인터넷의 발전은 데이터 전송 속도의 고속화, 대용량의 데이터 전송 등을 가져와 기업의 업무 효율을 향상시키고 전자상거래와 같은 서비스로 인하여 누구나 편리하게 사용되는 긍정적인 효과를 거두고 있는 반면 외부인의 시스템 불법 침입, 중요 정보 유출 및 변경, 훼손, 불법적인 사용, 컴퓨터 바이러스 및 네트워크를 통한 패킷 가로채기, 변조, 위조, 사칭과 같은 역기능들이 날로 증대되어 피해 규모가 심각한 수준에 이르고 있다. 특히, 전자 상거래와 같은 웹을 통한 데이터 전송에서 사용자의 신상정보 및 금융 정보들에 대한 보안 문제가 더욱 고려되고 있다. 이러한 대응책으로 암호화 및 복호화 기술과 메시지의 무결성, 사용자 인증과 기술이 개발되어 발전되고 있다. 본 논문에서는 이러한 기술을 응용하여 Linux Apache 웹 서버에 보안 모듈을 추가하여 웹 데이터의 암호화 및 무결성 서비스를 구현하고 클라이언트에서는 서버에서 받은 암호화된 메시지를 복호화 및 무결성 서비스를 구현한다.

본 논문의 구성은 다음과 같다. 2장에서는 관련연구를 살펴보고 3장에서는 서버/클라이언트 보안 모듈 설계 및 구현에 대하여 살펴보고 4장에서는 결론 및 향후 개선과제를 제시한다.

2. 관련 연구

2.1 Apache Module

Apache는 1995 년 그 당시에 가장 인기 있었던 웹 서버 중의 하나인 NCSA HTTPD 1.3 버전을 기반으로 탄생하였다. 그 후 기존의 NCSA 웹 서버에 더욱 향상된 기능들을 탑재하여 Apache 웹 서버를 발표하였다.

Apache에 사용자가 원하는 기능을 추가하기 위해서는 모듈 API를 이용한다. Apache에서 제공하는 API는 Perl과 C 언어의 형태로 제공한다.

Apache에 모듈을 추가하기 위해서는 다음의 과정을 수행한다.

(1) ~www/src/module/site에 사용자의 모듈의 소스파일을 추가

(2) Apache 컴파일 시

```
-./configure -□ activate-module =
                        src/modules/site/mod_write.c
- make
-make install
```

(3) httpd.conf 환경설정 파일 수정

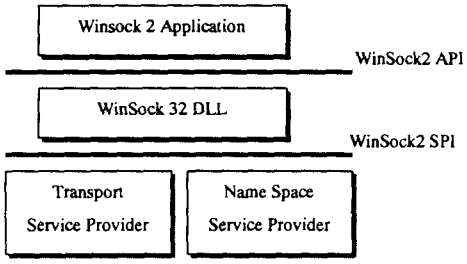
위의 과정을 통하여 Apache 웹 서버에서 동작되는 사용자 모듈을 추가할 수 있다.

2.2 Winsock 2

Winsock2는 Winsock 1.1 버전과의 완전한 호환성을 제공할 뿐만 아니라 다음과 같은 기능들이 확정되었다.

- TCP/IP 이외의 다른 프로토콜에도 접근 가능
- 프로토콜과 무관한 Name Resolution 능력
- Quality of Service
- 프로토콜과 무관한 Multicast, Multi-Point

WinSock2는 WOSA(Windows Open Services Architecture) 컴포넌트로 제작되었다. WOSA 는 어플리케이션과 서비스 간의 공통적인 인터페이스를 제공한다. 복잡한 시스템 내부에 대하여 신경 쓸 필요가 없고, 나중에 그 기능이 향상되거나 추가 된다고 하더라도 이전의 프로그램을 고칠 필요가 없다.



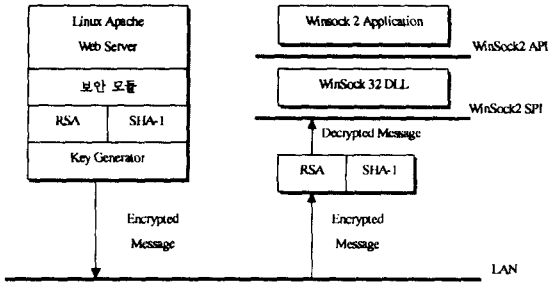
[그림1] Winsock 2 Overview

3. 서버/클라이언트 보안 모듈 설계 및 구현

3.1 서버/클라이언트 보안 모듈 설계

첫째, Apache 웹 서버에서 동작되는 보안 모듈은 다음의 기능을 갖는다. 서버에서 클라이언트로 전송되는 모든 메시지에 대한 암호화를 적용하기 위한 RSA 암호화 알고리즘, 메시지가 전송 도중 변경되지 않음을 증명하기 위한 SHA -1 해쉬 알고리즘, public/private키의 생성 기능이다.

클라이언트는 서버에서 보낸 암호화된 메시지를 해독하기 위한 RSA 복호화 알고리즘과 SHA-1 해쉬 알고리즘을 이용하여 전송도중에 변경임 없음을 확인할 수 있다. 서버와 클라이언트의 동작 과정은 다음과 같다.



[그림2] 서버/클라이언트 구조

클라이언트에서 요청이 발생하면 서버에서는 클라이언트의 요청에 따라서 해당 웹 페이지의 내용을 RSA 암호화 알고리즘을 이용하여 암호화한다. 암호화된 메시지는 클라이언트로 전송된다. 클라이언트에서 LSP보안 모듈은 암호화된 메시지를 RSA복호화 알고리즘으로 복호화한다. 복호화 된 메시지는 웹 브라우저에 정상적인 메시지 형태로 보여진다.

3.2 서버/클라이언트 보안 모듈 구현

3.2.1 서버 보안 모듈 구현

서버의 보안모듈을 구현하기 위해서 본 논문에서는 Apache 1.3.6버전을 사용한다. Apache 웹 서버에 보안 기능을 추가하기 위해서는 모듈의 형태로 추가하고 삭제하여야 한다. Apache에서 모듈을 구현하기 위한 C/Perl API 함수를 제공한다. 여기에서는 C API 함수를 이용한다. C API 함수는 클라이언트에 대한 request/response에 대한 함수가 있다.보안 모듈 기능은 구현하기위해 웹 서버에 클라이언트의 요청이 발생하는 웹 페이지에 대한 암호화를 수행한다. 그리고 SHA-1 해쉬 알고리즘을 이용하여 해쉬메시지를 생성한다. 모듈은 환경설정 파일의 설정에 따라 보안 모듈을 동작하도록 한다.

```
static int write_handler(request_rec *r)
{
      r->content_type = "text/plain";
      ap_send_http_header(r);

      if (r->header_only) {
            return OK;
      }
      write_fread(r);
      return OK;
}
```

[그림3] 보안 모듈 핸들러

보안 모듈은 핸들러는 클라이언트 요청에 대한 응답으로 HTTP 헤더의 content-type를 " text/html" 형태로 정의한다. 그리고 헤더가 성공적으로 전송되면 write_fread() 함수를 통하여 해당 웹 문서를 암호화를 수행한 후 암호화된 메시지를 클라이언트에 전송한다.

```
void *write_fread(request_rec *r)
{
      unsigned char buffer[1024];
      unsigned int len;
      int i;

      FILE *fh = ap_pfopen(r->pool, r->filename, "r");
      if (!fh) {
            return NULL;
      }
      rsa_data(r,fh);
      ap_pfclose(r->pool, fh)
}
```

[그림4] 메시지의 RSA암호화

3.2.2 클라이언트 보안 모듈 구현

클라이언트 보안 모듈은 Winsock 2 SPI 에서 제공하는 LSP(Layered Socket Protocol) 을 사용한다. SPI의 LSP 는 소켓과 채널을 연결하고 끊음으로써 사용자 임의로 보안에 대한 모듈을 설정할 수 있다. 소켓이 연결될 경우 보안 모듈이 동작하여 서버로부터 들어온 암호화된 메시지를 복호화 한 후 사용자의 웹 브라우저 즉 어플리케이션에 정상적인 메시지를 보여 준다. 그리고 보안 기능을 제공하는 웹사이트에 대한 데이터베이스를 구축하여 보안 기능에 제공되지 않은 일반 웹사이트를 탐색할 경우 URL 필터링 기능으로 보안 모듈이 동작하지 않는다. 이렇게 함으로서 선택적 보안 기능을 제공할 수 있다.

서버로부터 들어온 메시지는 크게 두 부분으로 처리할 수 있다. 첫번째 경우는 HTTP 헤더부분과 메시지가 같이 들어오는 경우이고 두 번째 경우는 메시지만 들어오는 경우이다. 첫번째 경우는 헤더부분에서 데이터의 내용이 "Content-Type:"인 위치부터 두개의 new line 을 찾아서 헤더와 메시지를 분리하여 메시지부분만 복호화하였다. 그리고 버퍼의 크기가 한정되어 메시지의 마지막 부분이 완전하게 들어오지 않고 일부분 들어오기 때문에 마지막 데이터를 파일로 저장하여 메시지만 들어온 경우의 첫번째 라인의 메시지와 붙임으로써 정상적인 메시지를 얻을 수 있었다.

```
while((pdest[cnt] != NULL)&&(cnt < MAX_BUFFER))
{
    for (i = 0; i < 20; i++)
        tmp1[i] = ' ';
    i = 0;
    cnt++;
    //
    // 한 라인 단위로 버퍼(tmp1)에 저장
    //
    while((i < 20) && (pdest[cnt] != '\n'))
    {
        tmp1[i] = pdest[cnt];
        i++;
        cnt++;
    }//while  ----------------------------------  φ
    decX = atol(tmp1);

    if (decX != 0)
    {
        DecodeChar[k]=decode(decX, T, N);
        strcpy(&string[length],DecodeChar);
        k++;
        strcpy(tmp2,tmp1);
    }//if  ----------------------------------  κ
    else
    {
        if(atol(tmp2) != 0)
        {
            fp = fopen("c:\\data.dat","w");
            fprintf(fp,"%s\n",tmp2);
            fclose(fp);
        } ----------------------------------  λ
    }
} //while
```

[그림5] LSP의 복호화 루틴

그림[5]의 ∽은 서버에서 보낸 암호화된 메시지를 한라인 단위로 구분한다. Tmp1변수에 한 라인의 값을 저장한다. ✱부분에서는 tmp에 저장된 값을 long 형태로 변환 후 복호화를 수행하고 수행된 결과를 리턴함으로서 클라이언트의 웹 브라우저에 정상적인 메시지가 출력된다. ✦부분에서는 메시지의 제일 마지막 부분의 데이터는 버퍼의 크기가 한정되어 있으므로 일부분만 들어오게 된다. 그러므로 정상적인 메시지로 복호화하지 못하고 잘못된 형태로 복호화된다. 이러한 문제점을 방지하기 위해서 마지막의 데이터를 파일의 형태로 저장하고 다음 메시지의 첫번째 라인과 연결함으로써 정상적으로 복호화된 메시지를 얻을 수 있다.

4. 결론 및 향후 과제

본 논문에서는 웹 서버/클라이언트 환경에서 각각 보안 모듈을 설계 및 구현하였다. 서버의 경우, 기존의 Apache 웹 서버 보안 프로그램은 Apache 와 보안 프로그램을 재컴파일 후 설치하는 불편이 있지만 본 논문은 라이브러리 형태로 사용자 임의의 요구에 맞는 보안 모듈을 추가함으로써 Apach e 웹 서버 자체에 대한 수정 없이 환경설정 파일 만 수정함으로써 보안 기능을 수행한다. 클라이언트의 경우, 사용자가 LSP 보안 모듈을 자유롭게 삽입 또는 삭제할 수 있으므로 사용자의 편의성을 향상시키는 방안을 제시하였다.

향후 과제는 RSA 암호화 알고리즘과 SHA-1에 사용되는 키가 고정되어 있으므로 랜덤한 키 생성 부분과 SHA-1를 통한 무결성 확인에 대한 처리가 개선되어야 한다

5. 참고 문헌

[1] Bob Quinn, Dave Shute, "Windows Sockets Network Programming", Addison-Wesley ,1995

[2] Pat Bonner, Patrick Bonner, "Network Programming with Windows Sockets", Prentice Hall, 1995

[3] Wei Hua, Jim Ohlund, Barry Butterklee , "Unraveling the Mysteries of Writing a Winsock 2 Layered Service Provider "http://www.microsoft.com/msj, 1999

[4] Mohammed J. Kabir, "Apache Server Bible", IDG Books Worldwide,1998

[5] Lincoln Stein, Doug MacEachern, Linda Mui, "Writing Apache Modules with Perl and C: The Apache API and mod_perl", O'Reilly & Associates, 1999

[6] S. C. Coutinho, "The Mathematics of Ciphers: Number Theory and RSA Cryptography", A K Peters, Ltd, 1998

[7] Michael Rosing, "Implementing Elliptic Curve Cryptography", Manning Publications Company;, 1998

[8] Jan C.A. Van Der Lubbe, "Basic Methods of Cryptography", Cambridge Univ, 1998