Apache 웹서버 보안 관리

2002. 10. 21

정현철, hcjung@certcc.or.kr

최근 침입차단시스템, 사용자 인증 시스템 등 각종 보안장비의 도입으로 인해 내부 시스템들은 보호되고 있지만, 웹서버는 대중에 공개되어야 하는 특성으로 인해 보안장비의 보호를 받지 못한 채 공격의 주요 목표가 되고 있다.

본 고에서는 웹서버 중 가장 많이 사용되고 있는 Apache 웹서버의 설치·운영 과정에서 고려하여야 할 보안 사항을 알아보도록 한다.

제 2장에서는 Apache 사용자 계정, 접근권한 등 초기 설치시에 고려하여야 할 사항들을 소개하고, 제3장에서는 환경설정을 통하여 디렉토리 리스팅 방지, 헤더정보 숨김, 에러메시지 수정 등 보안강화 방안에 대해 알아보고, 제4장에서는 ID와 패스워드에 의한 사용자 인증과 IP 주소에 의한 접근통제로 웹서버를 보호하는 방법에 대해 알아보고, 제5장에서는 로그관리, 보안패치, 데이터 백업 등 일상적인 운영을 위한 항목들을 살펴보기로 한다.

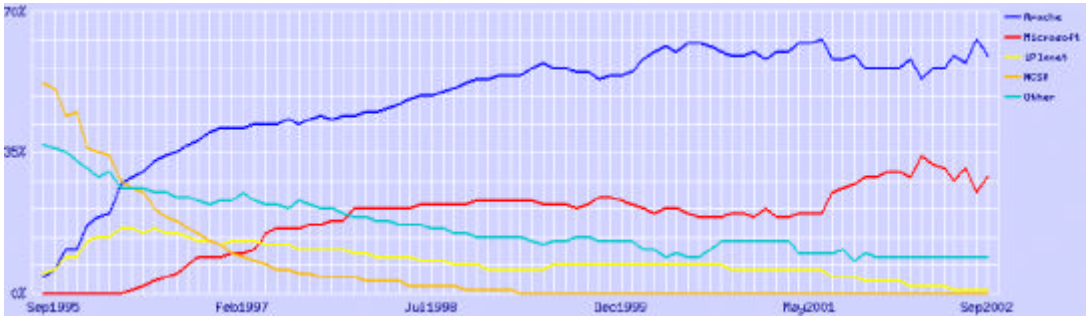
1. 개요

최근 침입차단시스템, 사용자 인증 시스템 등 각종 보안장비의 도입으로 인해 내부 네트워크 및 서비스들은 외부에서의 접근을 통제시키고 있다. 하지만, 다양한 인터넷 서비스 중 가장 많이 사용되고 있는 웹서비스 만큼은 대중에 공개되어야 하는 특성으로 인해 보안장비의 보호를 받고 있지 못하다.

이러한 이유에서 공격자들은 외부에 노출되어 있는 웹서버를 주된 공격 대상으로 삼는 것이 일반적이다. 웹 서버의 용도가 예전의 단순 홈페이지에서 인터넷 뱅킹, 민원 서비스 등 다양한 용도로 이용되고 있어 중요도가 증가하고 있지만, Apache OpenSSL 취약점 등 웹서버 자체의 취약점, 웹서버 설정상의 문제점, 그리고 웹 에플리케이션 및 웹 문서 작성시의 문제점 등 공격에 이용될 수 있는 보안 취약점은 대단히 다양하다.

웹서비스를 제공하는 웹서버 중 가장 많이 사용되고 있는 웹서버가 Apache 웹서버이다.

미국의 조사 전문기관인 넷크래프트(http:// / www.netcraft.com/ survey/)에 따르면 2002년 9월 현재 Apache 웹서버가 약 2,142만여대로 전체 웹서버 시장의 59.91%를 차지하고 있다고 발표하였다.



본 고에서는 Apache 웹서버를 설치하고 운영하는 과정에서 고려하여야 할 보안사항에 대해서 살펴보기로 한다. 현재 Apache 2.0.x 버전이 나와 있지만 국내에서 가장 많이 사용되고 있는 Apache 1.3.x를 기준으로 설명하기 위하여 리눅스 7.1 상에서 Apache 1.3.26을 설치하여 테스트했다.

2. Apache 설치시 보안고려 사항

가. 호스트 자체 보안

Apache 웹서버의 설치에 앞서 웹서버가 설치될 운영체제 자체의 보안이 당연히 우선시 되어야만 한다. 운영체제 보안은 본 고에서 언급하기에는 너무 방대할 수 있으므로 기본적인 사항만을 언급하기로 한다.

우선, 웹서버가 설치될 시스템은 웹서비스 이외의 다른 서비스 즉, NFS, FTP, telnet, mail 등의 부가적인 서비스를 제공하지 않고 웹서버 전용으로 운영하는 것이 바람직하다. 뿐만 아니라 서버에는 컴파일러나 소프트웨어 개발 도구 등 불필요한 도구들도 제거하는 것이 안전하다. 웹서버의 관리를 위해서는 Telnet이나 FTP 보다는 보안성이 뛰어난 SSH를 이용하는 것이 바람직하다. 그리고 서버상에 구동되는 모든 프로그램들은 보안 버그가 패치된 최신 버전을 유지하도록 한다.

호스트 자체의 보안을 위한 상세한 가이드는 다음 문서를 참고하기 바란다.

리눅스 시스템 관리자를 위한 보안 지침 I :
http://www.certcc.or.kr/paper/tr2001/tr2001-03/guide%20for%20Linux%20system%20admin.html

또한, 네트워크 차원에서 적절한 접근통제와 침입에 대한 모니터링이 필요하다.

라우터 보안 위협요소와 보안대책 :
http://www.certcc.or.kr/paper/tr2002/tr2002_06/020603-router_security.pdf

IpFilter를 사용한 Securing Solaris운영 :
http://www.certcc.or.kr/tools/firewall/IPfilter/IPfilter.html

Snort 설치 및 운영 가이드 :
http://www.certcc.or.kr/tools/Snort.html

나. Apache를 위한 사용자 계정 생성

Apache 웹서버 운영시에 위험을 최소화하기 위해서 웹서버 애플리케이션만을 위한 하나의 유일하고 관리자 권한이 아닌 최소 권한의 사용자 ID와 그룹을 생성하는 것이 안전하다. 대부분의 경우에 이러한 목적을 위해 사용자 ID와 그룹으로 "nobody"

를 사용한다. 물론 웹서버를 위해 nobody가 아닌 다른 root 권한 이외의 사용자 ID와 그룹을 생성해도 무방하다.

웹서버를 위해 생성된 root가 아닌 계정은 인터렉티브한 로그인을 허락해서는 않된다. 즉 이 계정을 통해서 사용자가 로그인할 수 없도록 해야 한다. 패스워드 파일에서 Apache를 위한 사용자 계정(nobody)이 locked되어 로그인할 수 없는지 확인한다.

```
</etc/passwd>
nobody:x:99:99:Nobody://

</etc/shadow>
nobody:*:11900:0:99999:7:::
```

Apache 서버가 초기설정될 때, 로그 파일을 열고, 웹서버 데몬을 구동하고, 적당한 TCP 포트(일반적으로 80 포트)를 열기 위해 root 권한이 필요하다. 일단 초기 구동 단계가 완료되면 웹서버 데몬은 root가 아닌 이미 사전에 생성한 사용자 ID와 그룹(nobody)으로 전환한다.

웹서버 데몬(httpd)이 root로 구동된 후 다른 사용자로 변환할 때 사용할 사용자와 그룹은 httpd.conf 파일에서 지정해 줄 수 있다. 일반적으로 nobody를 사용할 경우 다음과 같다.

```
User nobody
Group nobody
```

위의 설정으로 구동된 웹서비스 프로세스들은 초기 구동된 httpd 데몬 1개만이 root 권한이고 나머지는 nobody 권한으로 띄워져 있는 것을 확인할 수 있다. 일반 사용자들이 웹 브라우져를 이용해서 웹 서비스를 받을 경우 nobody 권한의 httpd 데몬과 연결을 맺게 된다.

```
#ps -aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root       968  0.0  0.8   2360  1036 ?        S    09:45   0:00 /usr/local/apache/bin/httpd
nobody     978  0.0  0.8   2512  1080 ?        S    09:45   0:00 /usr/local/apache/bin/httpd
nobody     979  0.0  0.8   2512  1080 ?        S    09:45   0:00 /usr/local/apache/bin/httpd
nobody     980  0.0  0.8   2512  1080 ?        S    09:45   0:00 /usr/local/apache/bin/httpd
nobody     981  0.0  0.8   2512  1080 ?        S    09:45   0:00 /usr/local/apache/bin/httpd
nobody     982  0.0  0.8   2512  1080 ?        S    09:45   0:00 /usr/local/apache/bin/httpd
```

일반적으로 standalone 형태의 웹서버가 리슨 하는 포트 번호가 80번인데, 포트 번호도 변경할 수 있다. 단, 1023 이하의 포트는 번호를 사용할 경우 처음 httpd를 구동시 반드시 root 권한이어야만 한다. 웹서버가 리슨하는 포트번호는 httpd.conf 파일에서 다음과 같이 지정할 수 있다.

```
Port 80
```

다. Apache 서버 소프트웨어 설치

웹서버를 안전하게 하기 위해서 서버 데몬과 컨텐트가 별도의 하드 디스크 파티션에 인스톨되어져야만 한다. 웹 컨텐트는 DocumentRoot에 설치되는데 DocumentRoot는 모든 웹 컨텐트가 저장될 Apache 내부의 디렉토리 구조이다. 가능하면 이 디렉토리는 서버 root나 chroot로부터 별도의 파티션이나 하드 드라이버를 사용하는 것이 좋다. Apache 웹서버 애플리케이션 파일들로부터 컨텐트를 분리하는 것은 공격자가 웹서버 공격을 훨씬 어렵게 한다.

Apache 기본 설치시에는 htdocs 디렉토리를 DocumentRoot로 사용하고 있는데 이를 바꾸도록 한다. htdocs 디렉토리는 공개되어서는 안될(또는 공개될 필요가 없는) Apache 문서를 포함하고 있으며, 공격에 이용될 수 있는 시스템 관련 정보도 포함하고 있다.

본 고에서는 /usr/local/www을 DocumentRoot로 지정하였다. 설정은 httpd.conf 파일에서 다음과 같이 할 수 있다.

```
#DocumentRoot "/usr/local/apache/htdocs"
DocumentRoot "/usr/local/www"
```

웹서버 데몬은 chroot에 설치하는 것이 안전하다. 만약 웹서버 데몬이 공격당했다고 하더라도 공격자는 chroot 디렉토리 이외로는 접근할 수 없어 피해를 최소화할 수 있다.

Apache 배포판의 src 디렉토리는 불필요한 실행파일을 컴파일할 수 있는 소스 코드들이 있어 서버에 남겨두지 않는 것이 안전하다.

라. 디렉토리와 파일의 접근권한

root에 의해 실행가능한 모든 명령어들은 항상 보호되어야 한다. root가 실행하는 명령어들이 root가 아닌 사용자들에 의해 변경되어서는 안된다. 따라서 이들 프로그램은 root에 의해서만 쓰기 가능하도록 하여야만 한다. 만일 root가 아닌 일반 사용자가 root가 실행하거나 쓰는 파일들에 대한 수정이 가능하다면 수많은 취약점과 공격에 노출되게 된다. 일반적으로 Apache 웹서버는 root로 시작되어지고 Apache 설정에서 정의된 사용자(nobody)로 전환된다. 이는 웹 페이지를 서비스하는 동안에 공격당하더라도 root 권한을 도용당하지 않도록 막아준다.

다음은 웹서버 설치시 생성되는 주요 하위 디렉토리 및 그 기능이다.

이름	기능	파일
conf	웹서버 설정	httpd.conf, srm.conf, access.conf
logs	웹서버 로그	access_log, error_log
cgi-bin	CGI 실행파일들	CGI 스크립트들(printenv, test-cgi 등)
bin	실행파일들	ab, apxs, dbmmanage, htpasswd, logresolve, apachectl, checkgid, htdigest, httpd, rotatelogs
htdocs	웹문서들	국가별 초기화면 및 메뉴얼
include	헤더파일들	
man	man 페이지	
icons	아이콘들	
support	도구들	관리용 유틸리티들

각 디렉토리 및 Apache 웹서버의 실제 데몬인 httpd의 소유자와 접근권한이 아래와 같이 설정되어 있는지 확인한다.

```
cd /usr/local/apache
chown 0 . bin conf logs
chgrp 0 . bin conf logs
chmod 755 . bin conf logs

chown 0 /usr/local/apache/bin/httpd
chgrp 0 /usr/local/apache/bin/httpd
chmod 511 /usr/local/apache/bin/httpd
```

만일 root 만이 실행하거나 쓸 수 있는 파일을 root가 아닌 사용자가 수정할 수 있다면 시스템은 root 권한을 도용당할 수 있다. 가령 누군가 httpd 실행파일을 다른 것으로 대체(replace)하고, 웹서버 관리자가 다음에 이 파일을 실행한다면 의도하지 않은 임의의 코드가 실행될 수도 있다. 또한 logs 디렉토리가 root가 아닌 다른 사용자가 쓸 수 있다면 로그파일을 다른 시스템 파일에 심블릭 링크 걸어서 root가 이 파일에 임의의 데이터를 overwrite 하게 만들 수도 있다. 그리고, log 파일 내용도 임의로 수정이 가능할 것이다.

DocumnetRoot 상의 모든 파일들은 "nobody" 권한으로 구동하는 웹 서버에 의해서 읽혀 질 수 있어야만 한다. 또한 내부의 웹 문서 작성자들은 document root에 파일을 자유롭게 추가 또는 수정할 수 있어야만 한다. 따라서 **DocumentRoot 디렉토리와 그 하위 디렉토리들은 웹문서 작성자 소유주와 그룹에 속해야 하고, 모든 이가 읽을 수 있어야 하고(world readable), 그룹 소속의 사용자들이 쓰기 가능해야만 한다.**

예를들어 웹문서 작성자를 "webmaste"라 하고, 그룹은 "webwrite"라고 가정해 보자.

"webwrite" 그룹에는 웹 문서를 추가하고 변경할 필요가 있는 계정을 추가하도록 한다. webmaste와 hcjung라는 사용자가 웹문서를 작성한다고 할 경우 /etc/passwd 파일과 /etc/group 파일의 등록 상황은 다음과 같이 할 수 있다.

```
</etc/passwd>
hcjung:x:500:501::/home/hcjung:/bin/bash
webmaste:x:501:501::/usr/local/www:/bin/bash

</etc/group>
webwrite:x:501:webmaste,hcjung
```

DocumnetRoot 하의 허가권은 다음과 같다.

```
[root@hcjung www]# ls -al
합계 20
drwxrwxr-x    2 webmaste webwrite     4096 10월 10 13:23 .
drwxr-xr-x   14 root     root         4096 10월 10 11:26 ..
-rw-r--r--    1 webmaste webwrite      450 10월 10 11:48 index.html
-rw-r--r--    1 hcjung   webwrite     1333 10월 10 13:23 test.html
```

마. 불필요한 CGI 스크립트 제거

Apache 배포판에는 불필요한 CGI 스크립트들이 포함되어 있어 공격에 이용될 수 있다. 특히, Apache 초기 버전의 경우 phf.cgi, Count.cgi, php.cgi 등 많은 CGI 스크립트들이 제공되었다.

Apache 설치시 기본적으로 cgi-bin 디렉토리에 설치되는 모든 CGI 스크립트들은 제거하는 것이 안전하다. 또한 필요에 의해 CGI 스크립트를 사용할 경우에는 사전에 그 안전성을 검토한 후에 설치하도록 한다. CGI 스크립트들과 다른 액티브 컨텐츠들은 웹서버 침해에 대단히 치명적인 잠재 위험을 내포하고 있다.

프로그래머가 고의로 보안 허점을 넣지 않았다고 하더라도 의도하지 않게 허점이 존재할 수도 있다. CGI 입력값은 반드시 검사후에 유닉스 명령어 라인에 전달되어야 한다. 이는 악의적인 사용자가 악성 코드나 명령어를 입력값으로 넣어 root 권한으로 실행되도록 하는 것을 막을 수 있다.

바. HTML 문서 트리로부터 모든 불필요한 파일들 제거

DocumnetRoot 디렉토리 내의 모든 불필요한 파일들은 제거되어져야 한다. 특히, 공개되어서 안될 정보를 포함한 파일들은 public하게 접근할 수 없도록 하여야 한다.

3. 환경설정

Apache 웹서버는 conf/httpd.conf 파일에서 웹서버 구동과 관련된 다양한 설정을 할 수 있다. 앞 장에서 잠시 살펴본 웹서버 포트 지정, 사용자 및 그룹 지정 등을 비롯해서 보안과 관련한 설정들도 여기서 제어할 수 있다. httpd.conf 수정 후에는 설정파일의 문법오류를 점검한 다음 변경된 내용을 반영하기 위해 웹 데몬을 재 가동하여야 한다.

```
[root@acjung bin] /usr/local/apache/bin/apachectl configtest
Syntax OK
[root@acjung bin] /usr/local/apache/bin/apachectl restart
/usr/local/apache/bin/apachectl restart: httpd restarted
```

본 장에서는 httpd.conf 설정파일에서 보안을 목적으로 고려해야 할 사항에 대해 알아보도록 한다.

가. 디렉토리 리스팅, 심블릭 링크, SSI 기능, CGI 실행

웹 서버에서 제공하는 각 디렉토리에서 고려해야 할 보안 관련 사항으로는 디렉토리 리스팅, 심블릭 링크, SSI 기능, CGI 실행 기능 제공여부를 검토하여야 한다.

앞으로 언급할 각 지시자에 의한 설정들은 <Directory>, <Location>, <Files> 등의 태크로 그 적용범위를 제한한다. 즉, <Directory /usr/local/www> ... </Directory> 의 경우 /usr/local/www 디렉토리에 대한 설정이다.

■ 디렉토리 리스팅

웹 브라우저에서 사용자가 URL을 입력했을 경우, Apache 웹서버는 3가지 방법 응답한다. 정상적으로 웹 내용을 보여주든지, 디렉토리 리스트를 보여주든지, 에러 메시지를 보여준다.

"아는 것이 힘이다"라는 속담이 있다.
원격의 공격자가 시스템에 대한 많은 정보를 획득할수록 보안 허점을 발견하기가 용이해 진다. 디렉토리 리스트를 보여주는 것 또한 불필요한 정보를 공격자에게 제

공하여 공격에 이용될 수 있다. 백업 데이터, CGI 소스코드들, 필요에 의해 만들어 놓은 심블릭 링크 등 서버 관리자가 실수로 지우지 않은 파일들이 공격자의 손에 들어갈 수 있다.

DocumentRoot 디렉토리 내의 모든 파일들이 리스팅되는 것을 방지하기 위해서 "Options" 지시자에서 "Indexes" 옵션을 제거하여야 한다.

■ 심블릭 링크

명몇 서버는 심블릭 링크를 이용해서 기존의 웹 문서 이외의 파일시스템에 접근 가능하도록 하고 있다. 이러한 방법은 편리할 수는 있지만 심각한 보안 문제를 야기시킬 수 있다. 가령 시스템 자체의 root 디렉토리(/)를 링크 걸게 되면 웹서버 구동 사용자 권한(nobody)으로 모든 파일시스템의 파일에 접근할 수 있게 된다. 즉, /etc/passwd와 같은 대단히 민감한 파일까지 누구나 열람가능하게 된다.
"Options" 지시자에서 심블릭 링크를 가능하게 하는 옵션인 "FollowSymLinks"를 제거함으로써 이를 막을 수 있다.

■ SSI(Server Side Includes)

SSI는 HTML 페이지 안에 위치하고 있으며, 동적인 웹 페이지를 제공할 수 있도록 한다. 하지만 SSI가 포함된 파일은 "exec cmd"를 사용해서 어떤 CGI 스크립트나 프로그램들을 Apache가 구동하는 사용자와 그룹 권한으로 실행시킬 수 있다.
이처럼 SSI 페이지가 스크립트나 프로그램을 실행시킬 수 없도록 하기 위해서는 "Options" 지시자에 "IncludesNoExec" 옵션을 추가함으로써 차단할 수 있다.

■ CGI 실행

사용자들이 CGI 스크립트들을 어느 디렉토리에서나 실행할 수 있도록 할 경우 악의적인 사용자가 CGI 프로그램을 업로드한 후 이를 실행하여 임의의 명령을 실행시킬 수 있다.
따라서, CGI 프로그램의 실행은 관리자가 지정한 특정 디렉토리에서만 가능하도록 제한할 필요가 있다. CGI 실행은 "ScriptsAlias" 지시자에 의해서 실행가능한 디렉토리를 제한할 수 있다. "ScriptsAlias" 지시자 문법은 다음과 같다.

```
Syntax: ScriptAlias URL-path file-path | directory-path
```

예를들어 cgi-bin이라는 디렉토리에서만 실행가능하도록 할 경우 다음과 같이 지정할 수 있다.

```
ScriptAlias /cgi-bin/ "/usr/local/apache/cgi-bin/"
```

앞서 언급한 디렉토리 리스팅, 심블릭 링크, SSI 등에 대한 제어는 "Options" 지시자에 의해 제어가 가능하다.

```
Syntax: Options [+|-]option [[+|-]option] ...
```

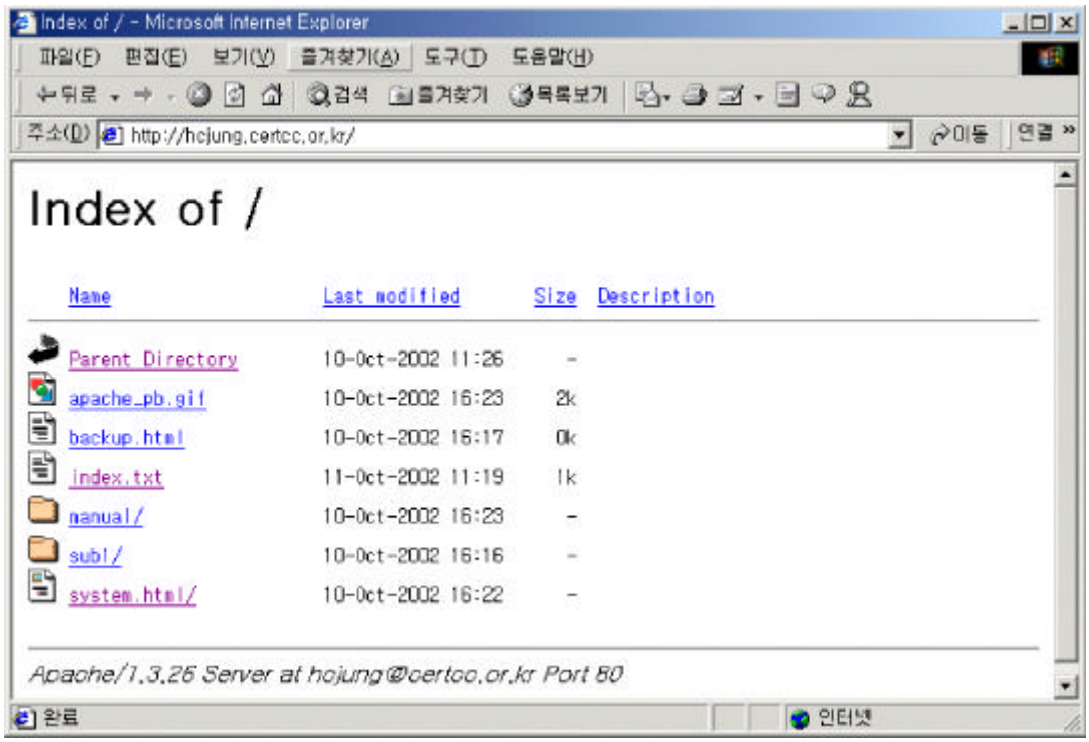
"Options" 지시자에서 사용할 수 있는 옵션값은 다음 표와 같다.

옵션값	설명
All	MultiViews를 제외한 모든 옵션을 줌(default 설정값임)
None	옵션을 주지 않음
ExecCGI	CGI 프로그램 실행을 가능하게 함
FollowSymLinks	심볼릭 링크로의 이동을 가능하게 함
Includes	Server Side Includes를 가능하게 함
IncludesNOEXEC	Server-side includes는 가능하지만 CGI 스크립트나 프로그램들은 실행할 수 없도록 함.
Indexes	해당 디렉토리 안에 DirectoryIndex에 명기된 파일(index.html 등)이 없을 경우 디렉토리와 파일 목록을 보여줌
MultiViews	유사한 파일이름을 찾아 주는 기능을 실행함(예를들어 index 라고만 입력하더라도 index.*를 찾아 보여줌)
SymLinksIfOwnerMatch	The server will only follow symbolic links for which the target file or directory is owned by the same user id as the link.

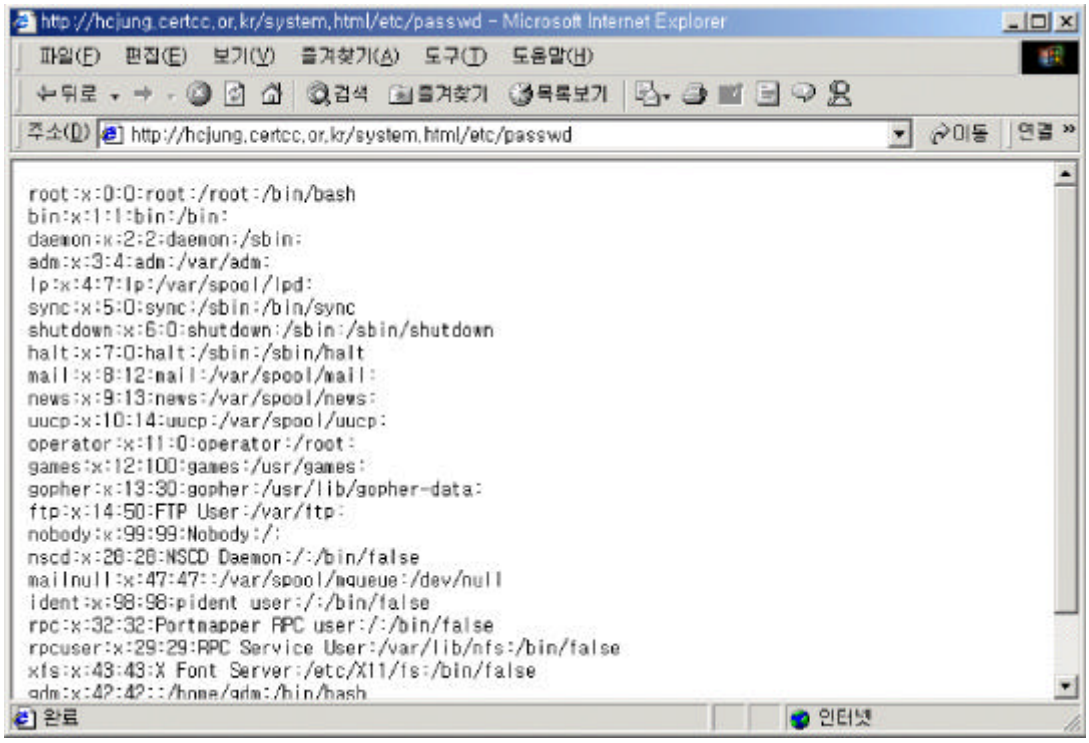
DocumentRoot 디렉토리에서 다음과 같이 설정되어 있다고 하자.

```
<Directory "/usr/local/www">
    Options Indexes FollowSymLinks
</Directory>
```

이 경우 다음 그림과 같이 DirectoryIndex에 정의된 초기 파일(index.html)이 존재하지 않을 경우 디렉토리내의 파일목록을 리스트업 해 준다.



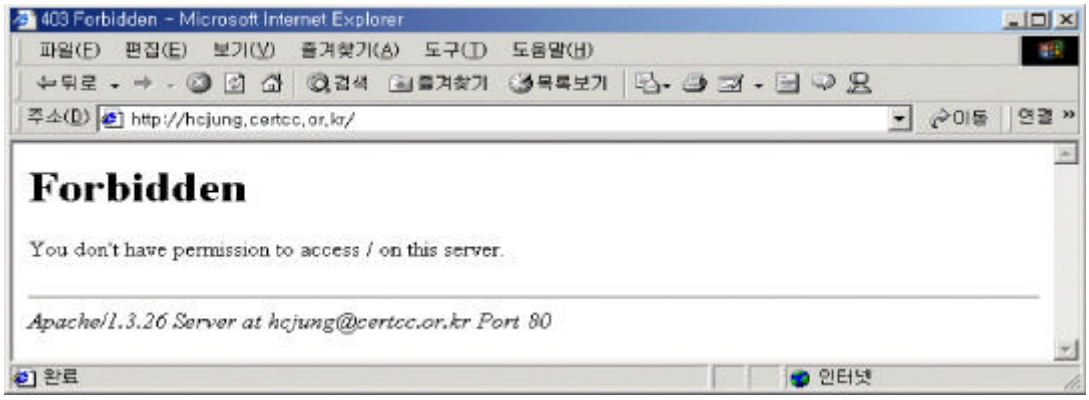
또한, FollowSymLinks로 인해 루트 디렉토리(/)에 심블릭 링크된 system.html 파일(ln -s / system.html)을 열었을 경우 DocumentRoot 디렉토리 상위의 passwd 파일까지 열람이 가능함을 알 수 있다.



이러한 문제점을 제거하기 위해서는 Indexes **옵션과** FollowSymLinks **옵션을 제거하고,** IncludesNoExec **옵션을 사용하도록 한다.**

```
<Directory "/usr/local/www">
    Options IncludesNoExec
</Directory>
```

이 경우 다음과 같이 초기 파일(index.html**)이 존재하지 않을 경우 디렉토리 리스트를 보여주는 것이 아니라 오류 창을 띄워주는 것을 확인할 수 있다.**



나. PUT과 POST의 제한

원격 사용자는 DocumentRoot 디렉토리 구조에 파일을 업로드 하거나 수정하는 행위가 제한되어야 한다. 물론 DocumentRoot의 파일/디렉토리 퍼미션을 사용해서도 웹을 통한 파일 업로드 및 수정을 막을 수는 있다. 하지만, 적절한 제한이 이루어지지 않을 경우 홈페이지가 변조되거나 웹 사이트가 침해당할 수 있으므로 <Limit> 태그를 이용하여 각 디렉토리별로 HTTP Method의 사용여부를 통제할 수 있다. 파일 업로드 및 파일의 수정을 위해서 사용되는 HTTP Method는 PUT과 POST이다.

다음의 예는 개인 사용자 홈디렉토리에서 POST, PUT, DELETE Method를 패스워드 파일에 등록된 사용자만이 이용가능하도록 제한한 것이다.(사용자 인증관련해서는 4장에서 설명할 예정임)

```
<Directory /home/*/public_html>
      <Limit POST PUT DELETE>
            Require valid-user
      </Limit>
</Directory>
```

다. 헤더 정보 숨김

클라이언트가 Apache 웹서버에 접속했을 때 웹서버에서는 응답 메시지의 헤더에 웹서버 버전, 설치된 응용프로그램 등과 같은 정보를 전달한다.

```
[root@acjung conf]# telnet xxx.xxx.xxx.xxx 80
Trying xxx.xxx.xxx.xxx...
Connected to xxx.xxx.xxx.xxx.
Escape character is '^]'.
GET / HTTP/1.1

HTTP/1.1 400 Bad Request
Date: Tue, 15 Oct 2002 11:25:10 GMT
Server: Apache/1.3.19 (Unix) PHP/4.0.4pl1
```

이 정보는 공격자에 의해 Apache 웹서버 버전별 또는 구동되고 있는 응용프로그램에 잘 알려진 취약점을 공격하는데 유용하게 이용될 수 있으며, 인터넷 웜과 같은 자동화된 공격에서도 이러한 banner 정보가 사용되어지기도 한다. 따라서 공격자에게 웹서버의 버전과 같은 banner 정보를 숨기는 것이 안전하다.

Apache 웹서버에서는 "ServerTokens" 지시자를 수정함으로써 헤더에 의해 전송되는 정보를 바꿀 수 있다.

```
Syntax: ServerTokens Minimal|ProductOnly|OS|Full
```

ServerTokens 지시자에서 설정할 수 있는 각 키워드에 의해 나타나는 정보는 다음과 같다.

키워드	제공하는 정보	예
Prod[uctOnly]	웹서버 종류	Server: Apache
Min[imal]	Prod 키워드 제공 정보 + 웹서버 버전	Server: Apache/1.3.0
OS	Min 키워드 제공 정보 + 운영체제	Server: Apache/1.3.0 (Unix)
Full	OS 키워드 제공 정보 + 설치된 모듈(응용프로그램) 정보	Server: Apache/1.3.0 (Unix) PHP/3.0 MyMod/1.2

ServerToken은 Apache 1.3 이상에서 가능하고 ProductOnly 키워드는 1.3.12 이상의 버전에서 사용가능하다. 일반적으로 ServerToken은 httpd.conf에서 명시되어 있지 않는 경우가 많은데 이럴 경우에는 기본값인 "ServerTokens Full"이 적용되어 운영체제 정보, 웹서버 정보, 버전정보, 설치된 모듈 정보 등이 모두 서버의 응답 헤더에 포함되어 클라이언트에게 전송된다.

최소한의 정보를 주기 위해서 "ServerTokens Prod"를 설정하는 것이 바람직하다.

혹자는 공격자를 속이기 위해서 서버의 헤더 정보를 앞에서 명기한 내용과는 전혀 다른 내용으로 조작하여 클라이언트에 보내기를 원할 수도 있는데 이 경우에는 Apache 소스코드를 수정해서 재 컴파일하여야 하므로 권고하지 않는다.

라. 에러 메시지 수정

웹서버 구동중에 문제나 에러가 발생되면 Apache는 다음의 4가지 중 1가지 행위를 하게 된다.

- 간단한 시스템에서 작성된 에러 메시지 출력
- 사용자가 수정한 메시지 출력
- 문제나 에러를 해결하기 위한 로컬 URL을 리다이렉션
- 문제나 에러를 해결하기 위한 외부 URL을 리다이렉션

이 중 첫 번째가 기본적인 설정이며 나머지를 사용하기 위해서는 "ErrorDocument" 지시자를 사용하여 수정해야 한다.

"ErrorDocument" 지시자의 문법은 다음과 같다.

```
Syntax: ErrorDocument error-code document
```

여기서 error-code는 HTTP 규약(RFC2616)의 10번째 세션 "상태코드 정의"에 자세히 설명되어 있다. (http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html)

분류	상태코드	설명	분류	상태코드	설명
1xx : Informational	100	Continue	4xx : Client Error	404	Not Found
	101	Switching Protocols		405	Method Not Allowed
2xx : Successful	200	OK		406	Not Acceptable
	201	Created		407	Proxy Authentication Require
	202	Accepted		408	Request Time-out
	203	Non-Authoritative Information		409	Conflict
	204	No Content		410	Gone
	205	Reset Content		411	Length Required
	206	Partial Content		412	Precondition Failed
3xx : Redirection	300	Multiple Choices		413	Request Entity Too Large
	301	Moved Permanently		414	Request-URI Too Large
	302	Moved Temporarily		415	Unsupported Media Type
	303	See Other	5xx : Server Error	500	Internal Server Error
	304	Not Modified		501	Not Implemented
	305	Use Proxy		502	Bad Gateway
4xx : Client Error	400	Bad Request		503	Service Unavailable
	401	Unauthorized		504	Gateway Time-out
	402	Payment Required		505	HTTP Version not supported
	403	Forbidden			

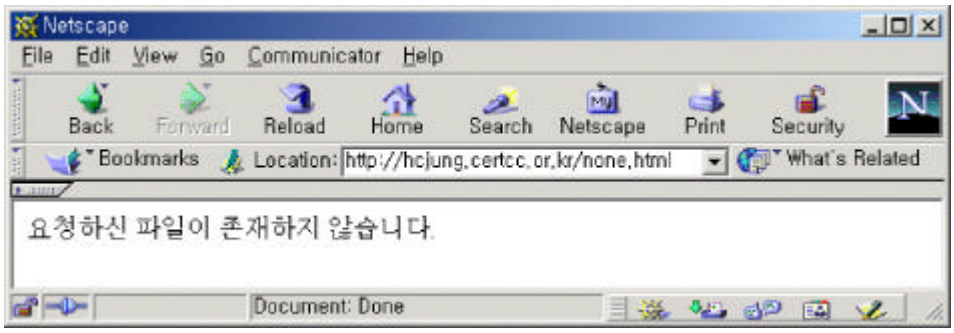
다음은 "ErrorDocument" 지시자의 사용 예인데 직관적으로 알 수 있을 것이다.

```
ErrorDocument 401 /subscription_info.html
ErrorDocument 403 http://dci.sppo.go.kr/
ErrorDocument 404 "요청하신 파일이 존재하지 않습니다.
```

첫 번째 예는 사용자 미등록 등으로 인해 사용자 인증 실패시 사용자 등록 페이지를 안내하고 있다.
두 번째 예는 접속이 허락되지 않은 IP에서 접속을 시도할 경우 대검찰청 인터넷범죄수사센터로 리다이렉션되도록 하였다. 이 경우 access_log에는 403 상태코드(Forbidden) 대신 302 상태코드(Moved Temporarily)가 기록되게 된다.

```
172.16.5.17 - - [17/Oct/2002:10:12:56 +0900] "GET / HTTP/1.1" 302 290
```

세 번째 예는 404 상태코드(not Found)의 경우 아래와 같이 클라이언트의 웹 브라우저에 "요청하신 파일이 존재하지 않습니다."라는 문구를 출력하도록 하였다.



이처럼 사용자가 에러 메시지를 수정할 경우 "ErrorDocument" 지시자에 하나의 쌍따옴표(") 만을 사용하여 문장을 넣어야함에 주의해야 한다.

4. 사용자 인증 및 접근통제

접근을 허락하기에 앞서 사용자나 호스트 인증과 접근통제를 위한 몇몇 기능이 Apache에 있다. 이는 특정한 IP 주소나 서브넷에 따라서 접속을 허락하거나 거부할 수 있고, 사용자 이름과 패스워드에 의해서 사용자를 인증할 수도 있다.

가. 사용자 인증의 종류

■ 기본 사용자 인증

HTTP는 stateless 프로토콜이므로 기본적인 사용자인증에 의해 보호되는 자원에 접근하기 위해서는 매번 사용자 이름과 패스워드와 같은 인증서를 서버에 보내야만 한다. 하지만 초기 인증을 거친 후 다른 페이지에 접근하기 위해서 매번 사용자 이름과 패스워드를 서버에 전송하는 것은 일반적으로 클라이언트 소프트웨어나 웹 브라우져에 의해서 자동으로 이루어진다. 만약 사용자 이름이 웹서버의 리스트에 있고, 패스워드가 일치하면 보호된 자원에 접근을 허락받게 된다.

기본적인 인증에서는 패스워드가 암호화되어서 저장되지만 클라이언트에서 서버로 전송되는 도중에는 암호화되지 않아 제 3자에 의해서 도청될 수 있다. 보호된 자원에 접속하는 매 순간마다 ID와 패스워드가 전송되므로 telnet, ftp 등 인증을 하는 다른 서비스보다 쉽게 도청이 가능하다. 뿐만 아니라 서버에서 클라이언트로 전송되는 어떠한 데이터에 대해서도 암호화가 제공되지 않으므로 내용도 가로채기가 용이하다. 따라서 기밀성이 중요시되는 웹서버에서는 이러한 인증은 권장할 수 없다.

■ 다이제스트 사용자 인증

두 번째 인증 방법으로는 다이제스트 인증이 있는데 기본적인 인증과의 차이점은 네트워크 등 전송로 상에서 패스워드가 평문으로 전송되지 않는다는 점이다. 패스워드는 MD5 암호화 해쉬를 시킨 후 전송한다. 다이제스트 인증은 패스워드를 암호화해서 전송하고는 있지만 데이터는 평문으로 전송되므로 문제점을 가지고 있고, 또한 모든 웹브라우져가 다이제스트 인증을 지원하지는 않는다는 문제점이 있다.

■ DB 인증 모듈

DB 인증 모듈은 사용자 이름과 패스워드를 보다 신속하게 확인할 수 있도록 한다. 서버에 다수의 사용자 이름과 패스워드가 저장되어 있을 경우 사용자가 데이터에

접근하기 위한 인증과정에 많은 시간이 소모될 수 있다. 일반 파일 시스템이 아닌 DB를 이용할 경우 사용자 이름과 패스워드 확인 시간을 대단히 단축할 수 있다.

나. 기본 사용자 인증 설정

기본 사용자 인증과 다이제스트 사용자 인증의 설정 방법은 대단히 유사한데 다음과 같은 절차를 거쳐서 설정할 수 있다.

- 패스워드 파일 생성
- 패스워드 파일을 사용할 수 있도록 Apache 환경 설정

본 고에서는 일반적으로 많이 사용되고 있는 기본 사용자 인증 설정 절차에 대해서 살펴보기로 한다.

① 패스워드 파일 생성

/usr/local/apache/bin/htpasswd 명령을 이용하여 패스워드 파일을 생성한다.
htpasswd 파일의 사용법은 다음과 같다.

```
Usage: htpasswd [-cmdps] passwordfile username
```

패스워드 파일을 최초로 생성할 경우에는 -c 옵션을 사용하여 새로운 패스워드 파일을 만든다.

```
[root@hcjung bin]# ./htpasswd -c /usr/local/apache/passwords hcjung
New password:
Re-type new password:
Adding password for user hcjung
```

이후 새로운 사용자를 추가하고자 할 경우에는 -c 옵션을 빼고 사용하면 된다. 실수로 -c 옵션을 줄 경우 기존에 등록된 사용자들이 지워지므로 주의하여야 한다.

```
[root@hcjung bin]# ./htpasswd /usr/local/apache/passwords webmaste
```

패스워드 파일은 사용자ID와 패스워드 필드로 구성되어 있는데 패스워드 필드는 암호화되어 저장된다.

```
[root@hcjung bin]# ls -al /usr/local/apache/passwords
-rw-r--r--       1 root         root                   45  10월  12  09:55
/usr/local/apache/passwords

[root@hcjung bin]# cat /usr/local/apache/passwords
hcjung:kLl/qVXc9xS7M
webmaste:lsMhrHQSYC1Ts
```

생성된 패스워드 파일은 가능한 안전한 장소에 보관하고 웹서버 자체가 읽을 수 있는 최소한의 권한만을 주어야만 한다. 만일 웹서버가 nobody 사용자와 nobody 그룹으로 구동된다면 다음과 같이 소유권과 접근권한을 줄 수 있다.

```
[root@hcjung bin]# chown root.nobody /usr/local/apache/passwords
[root@hcjung bin]# chmod 640 /usr/local/apache/passwords
```

② 패스워드 파일을 사용가능하도록 환경설정

패스워드 파일의 생성이 끝났으면 Apache 웹서버에게 이 파일을 사용할 수 있도록 설정하여 주어야 한다.
먼저 각 디렉토리별로 사용자 인증을 하기 위해서 httpd.conf 파일 내의 AllowOverride 지시자의 옵션을 None에서 AuthConfig 또는 All로 바꾼다.(사용자 인증만을 위해서는 AuthConfig 사용을 권고)

```
<Directory "/usr/local/www">
    AllowOverride AuthConfig
</Directory>
```

그리고, 사용자 인증이 필요한 디렉토리에 다음의 지시자들이 포함된 .htaccess 파일을 생성한다.

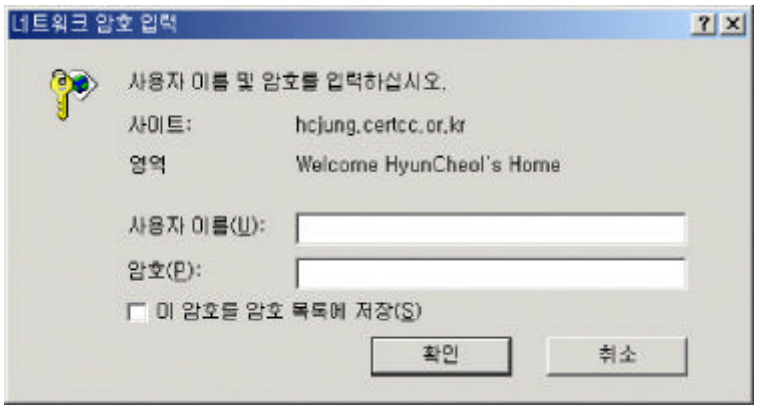
지시자	설명
AuthType	인증 형태(Basic 또는 Digest)
AuthName	인증 영역(웹 브라우저의 인증창에 표시됨)
AuthUserFile	사용자 패스워드 파일의 위치
AuthGroupFile	그룹 파일의 위치(옵션)
Require	접근을 허용할 사용자 또는 그룹 정의 ex) Require user userid [userid] ... Require group group-name [group-name] ... Require valid-user

앞서 패스워드 파일에 등록된 hcjung와 webmaste만이 웹서버에 접속할 수 있도록 하기 위해서는 다음과 같이 설정할 수 있다.

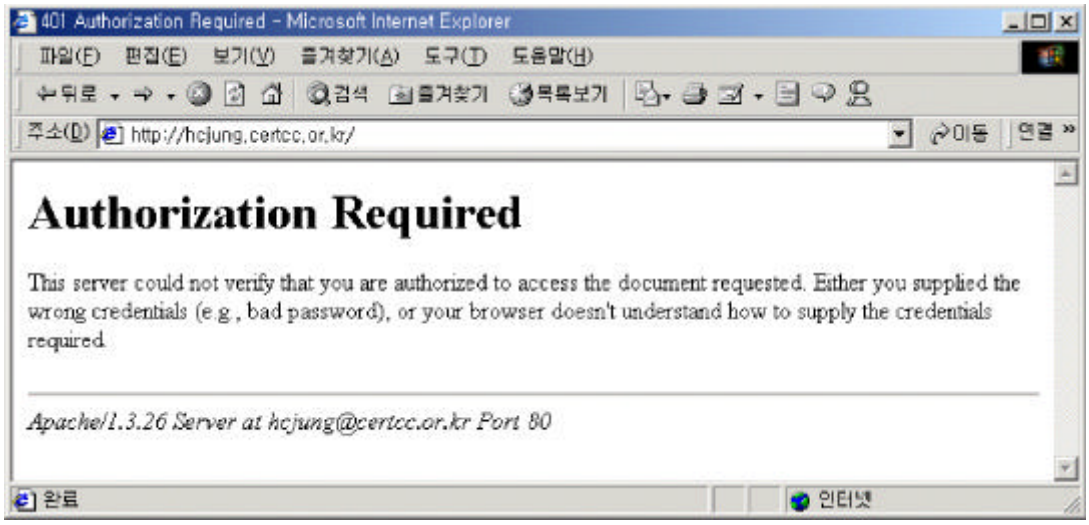
```
[root@hcjung /root]# cd /usr/local/www
[root@hcjung www]# vi .htaccess
AuthType Basic
AuthName "Welcome HyunCheol's Home"
AuthUserFile /usr/local/apache/passwords
Require user hcjung webmaste
```

위에서 접근을 허용할 사용자를 hcjung와 webmaste로 한정을 했는데 패스워드 파일에 등록된 모든 사용자들이 접근할 수 있도록 하기 위해서는 사용자를 지정하는 대신 "Require valid-user"라고 하면 된다.

정상적으로 사용자 인증 설정이 완료되었을 경우 웹 브라우져에서 웹서버 접속시 다음과 같은 사용자 이름과 암호를 묻는 인증창이 뜨게 된다.



사용자 이름과 암호가 정확하게 입력된 경우는 웹 페이지 접속이 가능하지만 정확하지 않을 경우 다음과 같은 경고창이 뜨고 접속을 허가하지 않는다.



다. 접근통제

클라이언트가 사용하는 호스트의 IP 주소나 도메인에 의해서 웹서버의 데이터에 대한 접근을 통제할 수 있다. 기본적인 서버 설정은 DocumentRoot의 내용에 대해 누구나 접속을 허락하도록 설정되어 있다.

Apache의 "Allow"와 "Deny" 지시자는 사용자 시스템의 호스트 이름과 호스트 주소를 근간으로 접속을 허락 또는 차단할 수 있도록 지정할 수 있다. 또한, "Allow"와 "Deny" 지시자를 동시에 사용할 경우 그 순서를 정하는 "Order" 지시자를 사용하여 보다 정교한 정책설정을 할 수 있다.

지시자	설명
Order Deny,Allow	Deny 지시자가 Allow 지시자 보다 먼저 검사된다. 접근은 기본적으로 허용된다. 즉, Deny 지시자나 Allow 지시자에 일치하지 않는 클라이언트의 접속은 허용한다.
Order Allow,Deny	Allow 지시자가 Deny 지시자 보다 먼저 검사된다. 접근은 기본적으로 차단된다. 즉, Deny 지시자나 Allow 지시자에 일치하지 않는 클라이언트의 접속은 차단한다.
Order Mutual-failure	Allow 리스트에 있고 Deny 리스트에 없는 호스트만 접근을 허용한다. 순서는 "Allow,Deny" 때와 같다.

일반적으로 Firewall이나 라우터의 접근통제 Rule은 순차적으로 비교하다가 최초로 일치하는 Rule을 적용하고 그 이후는 비교하지 않지만, Apache에서는 Allow와 Deny를 일단 모두 비교하고 둘 중에 하나라도 일치할 경우 적용한다는 점에서 차이가 있음을 생각해야 한다.

"Order" 지시자 사용시 키워드(Allow 또는 Deny)는 콤마(,)에 의해서만 분리되고 공백이 들어가서는 안 된다.

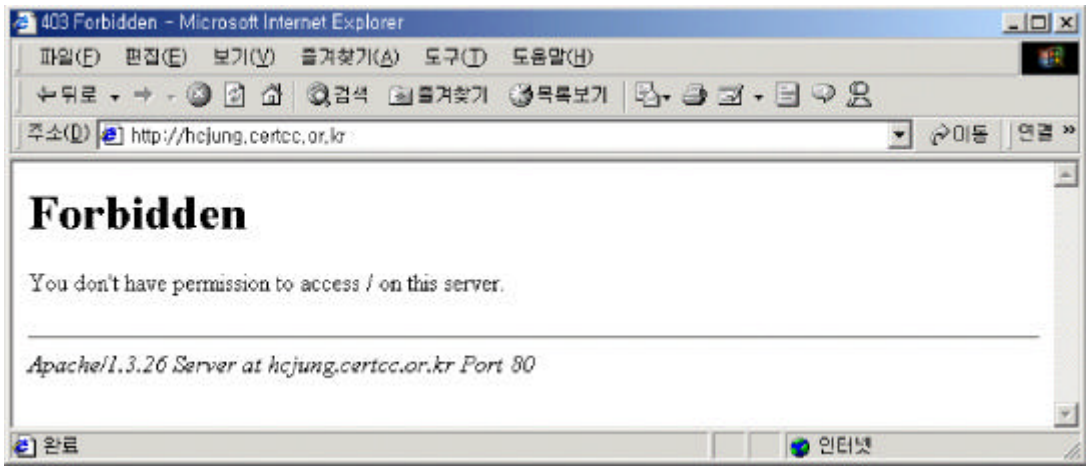
로컬 인트라넷(예를들어 172.16.5.0/ 24)에서만 웹서버의 컨텐츠에 접속하도록 하고 그 이외의 IP에서는 접속을 차단하기 위해서는 다음과 같이 설정할 수 있다.

```
order deny,allow
deny from all
allow from 172.16.5
```

"deny from"과 "allow from" 지시자는 호스트, 도메인 이름, IP 주소, 서브넷 마스크를 가진 IP 주소(예를들어 172.16.5.0/ 255.255.255.0), CIDR(Classless InterDomain Routing) 마스크를 가진 IP 주소(예를들어 172.16.5.0/ 24)를 사용할 수 있다.

다음은 접속을 허용하지 않는 주소에서의 접속 시도시 403 상태코드(Forbidden)를

출력하고 접속이 거부되는 것을 볼 수 있다.



라. 권한부여(Authorization)

권한부여는 특정한 자원에 접근할 사용자 퍼미션이 유효한지를 확인하는 과정이다. 어떤 퍼미션에 의해 허락되고 거부될지는 자원과 그 자원과 관련된 규칙들에 따라서 대단히 다양하다. 각 파일과 디렉토리 구조는 다른 접근통제나 사용자 인증 방법을 가질 수 있다. 접근통제와 사용자 인증 방법을 사용하여 각 자원에 대한 다양한 권한을 부여할 수 있다. 가령 인터넷에서 접속시에는 사용자 이름과 패스워드를 확인하고 인트라넷에서 접속시에는 요구하지 않도록 설정할 수도 있다. 이는 "Satisfy" 지시자를 통해서 구현할 수 있다.
즉, "Satisfy" 지시자는 사용자 인증(Require에 의한)과 클라이언트 호스트 주소에 따른 접근통제(Allow에 의한)를 동시에 사용하여 정책설정을 할 때 쓰인다.

```
Syntax: Satisfy any|all
```

다음 예는 인트라넷 밖에서의 모든 접속시 패스워드를 요구하고 인트라넷 내부의 사용자들은 패스워드 없이 접속을 허용하도록 설정한 예이다.

```
order deny,allow
deny from all
allow from 172.16.5
AuthType Basic
AuthName “"Welcome HyunCheol's Home"
AuthUserFile /usr/local/www/passwords
Require hcjung webmaste
Satisfy Any
```

만일 인트라넷 사용자인 동시에 이 사용자들에게 패스워드를 요구하게 강화된 보안을 설정할 경우 “Satisfy Any"를 ″Satisfy All"로 바꾸면 된다. 즉, ″Satisfy All"은 두 개의 지시자(접근통제와 사용자 인증) 모두 만족해야만 할 경우 사용하고, “Satisfy Any"는 둘 중 하나만 만족하면 된다.

마. SSL/TLS 인증

앞서 살펴본 사용자 인증기법들은 모두 웹 컨텐츠를 암호화하지는 않는다는 단점이 있었다. 최근 인터넷 뱅킹 등과 같이 전송로 상에 전송되는 웹 컨텐츠 역시 보호되어져야 하는 경우가 많다.

SSL/TLS는 사용자 인증과 웹서버 데이터와 컨텐츠를 암호화하는 훌륭한 수단이다. SSL을 지원하기 위해서 Apache는 Mod_SSL 모듈을 가지고 있고, 이 모듈은 SSL v2, v3 그리고 새로운 TLS을 사용하는 강력한 암호화를 제공한다. 현재 이 모듈은 강력한 128bit 암화화와 RSA와 Diffie-Hellman 암호화를 제공한다.

하지만, 최근 Apache OpenSSL 버퍼오버플로우 취약점을 이용한 Apache/mod-ssl 웜(또는 Slapper 웜으로도 불림)으로 인해 많은 Apache 웹서버들이 공격당하고 있으므로 OpenSSL 버전 0.9.6e을 사용하여야 한다.

다음은 SSL 프로토콜이 제공하는 주요 기능이다.

- 사설 접속과 데이터 암호화
- 서버에 통신하는 단말 인증
- 신뢰된 접속

최초 핸드쉐이크 후에 SSL은 비밀키를 생성하고 이 대칭키 암호화가 데이터 암호화를 위해 사용된다. 공개키(비대칭키)는 단말의 신원 인증과 대칭키 교환에 사용된다. 메시지 무결성은 MAC(Message Authentication Code)에 의해 제공되고 신뢰된 접속을 가능하게 한다.

5. 운영관리

가. 로그관리

Apache 로그파일들은 기본적으로 /usr/local/apache/logs 디렉토리에 저장된다. 기본 설치시에는 access_log와 error_log 2개의 로그파일이 생성된다.

```
[root@cjung /root]# cd /usr/local/apache/logs
[root@cjung logs]# ls -al
합계 72
drwxr-xr-x    2 root     root         4096 10월 15 17:33 .
drwxr-xr-x   12 root     root         4096 10월 12 11:17 ..
-rw-r--r--    1 root     root        31676 10월 16 09:56 access_log
-rw-r--r--    1 root     root        25234 10월 16 09:56 error_log
-rw-r--r--    1 root     root            4 10월 15 20:15 httpd.pid
```

(1) access_log

access_log는 서버에서 처리된 모든 요청들의 기록을 가지고 있다. access_log의 위치와 로그 포맷은 "CustomLog" 지시자에 의해 정의된다.

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog /usr/local/apache/logs/access_log common
```

위와 같이 기본 설정된 access_log 로그는 8개의 필드로 구성되어 있다.

- 클라이언트 IP 주소
- 유일한 개인 ID(클라이언트의 ident 값)
- 사용자 이름(사용자 인증을 거친 경우 사용자 이름)
- 날짜
- Method(GET, PUT, POST 등)
- URI(Uniform Resource Identifier)
- HTTP 상태
- 전송된 바이트 수

다음은 access_log의 예이다.

```
172.16.5.17 - - [17/Oct/2002:10:12:56 +0900] "GET / HTTP/1.1" 302 290
172.16.5.16 - hcjung [17/Oct/2002:10:58:38 +0900] "GET / HTTP/1.1" 304 -
172.16.5.16 - hcjung [17/Oct/2002:10:59:21 +0900] "GET / HTTP/1.1" 302 290
172.16.5.16 - hcjung [17/Oct/2002:11:00:43 +0900] "GET / HTTP/1.1" 403 286
172.16.5.16 - hcjung [17/Oct/2002:11:01:25 +0900] "GET /index.htm HTTP/1.1" 200 453
```

웹서버 로그는 운영체제 로그와 더불어서 주기적으로 검사하여야 한다. 검사시에 다음 사항들을 점검하도록 한다.

- 유효하지 않은 로그인 시도
- 제한된 필드에 대한 접속 시도
- 존재하지 않는 파일에 대한 접속 시도
- 허락되지 않은 PUT(업로드) 시도
- 단기간 동안의 동일한 IP 주소로 부터의 대량 접속 시도(서비스거부)
- 웹서버의 예기치 못한 종료와 시작

다음은 phf CGI 프로그램의 취약점을 이용하여 패스워드 파일을 열람한 기록이 access_log에 남아 있는 것을 확인할 수 있다.

```
xxx.xxx.xxx.xxx - - [16/Jun/1998:10:38:02 +0900]
"GET /cgi-bin/phf?Qname=root%0Acat%20/etc/passwd HTTP/1.1" 200 114873
```

또한, Apache 게시판에 서버를 해킹하겠다는 협박성 내용이 게시되었다는 신고를 접수하고 해당 웹서버의 access_log를 분석하여 해당 게시번호에 대한 PUT 또는 POST Method를 남긴 IP를 찾아 게시자를 추적한 사례도 있었다. 그리고, 지난 2001년에 전세계를 떠들썩하게 했던 CodeRed 웜의 공격 흔적도 access_log에 기록되었던 것을 기억할 것이다.

(2) error_log

Apache 웹서버의 진단 정보 및 요청 처리과정에서 발생되는 각종 에러에 대한 기록을 남기는 error_log는 "ErrorLog" 지시자에 의해서 위치와 이름이 정해진다. 비정상적인 웹서버의 종료 등과 같은 문제가 발생되었을 경우 가장 먼저 error_log의 분석이 필요하다.

본 고의 작성을 위해서 각종 설정을 변경하면서 발생되었던 문제들도 error_log를 통해 해결할 수 있었다. 설정을 변경하여 웹서버를 테스트할 때에는 다음 명령으로 error_log의 내용을 실시간으로 모니터링할 수 있었다.

```
[root@hcjung logs]# tail -f /usr/local/apache/logs/error_log

[Wed Oct 16 13:00:50 2002] [error] [client 172.16.5.16] user hcjung:
authentication failure for "/": password mismatch
[Wed Oct 16 13:01:12 2002] [error] [client 172.16.5.16] user hcjung89
not found: /
[Wed Oct 16 13:02:38 2002] [error] [client 172.16.5.16] File does not
exist: /usr/local/www/nonpage.html
```

error_log는 아래와 같이 4개의 필드로 구성되어 있다.

- 메시지의 날짜와 시간
- 보고된 에러의 심각성 수준
- 에러를 발생시킨 클라이언트의 IP 주소
- 에러 메시지 내용(대단히 다양함)

기본 설정상태의 httpd.conf에서는 error_log와 관련하여 다음과 같이 되어 있다.

```
ErrorLog /usr/local/apache/logs/error_log
LogLevel warn
```

"LogLevel" 지시자는 에러로그에 어느 정도 이상의 심각성이 있을 경우 로그를 남

길 것인지를 지정할 수 있는데, 일반적인 유닉스 시스템의 syslog와 마찬가지로 debug, info, notice, warn, error, crit, alert, emerg 중 하나를 선택할 수 있다. debug는 심각성이 가장 낮은 수준의 로깅이며 emerg는 심각성이 가장 높은 수준의 로깅을 의미한다. "LogLevel"을 debug로 지정할 경우 대단히 사소한 내용도 기록되므로 로그의 량이 대단히 방대해 질 수 있으므로 적정한 수준에서 기록하도록 하여야 한다.

나. 보안 패치

앞서 살펴본 웹 서버 관리자에 의한 환경 설정 상의 문제점으로 인한 공격 가능성과 더불어 웹서버 자체 또는 웹서버에서 사용하는 애플리케이션 프로그램의 버그로 인한 공격도 심각하다. 특히, 요즘 극성을 부리고 있는 인터넷 웜의 경우 각 서버에 공통적인 취약점을 찾아서 공격하므로 웹서버에 대한 보안 패치는 수시로 이루어져야만 한다.

최근 Chunked encoding 취약점, Open SSL 취약점 등 서비스 거부공격 뿐만 아니라 임의의 명령어 실행까지 가능한 다수의 취약점이 발견되었다.
버전별로 발견된 취약점은 ApacheWeek(http://www.apacheweek.com/security/)에서 확인할 수 있다.

웹서버의 보안 취약점은 Nessus, Whisker 등과 같은 공개 취약점 점검 도구나 상용 취약점 분석 도구를 이용하여 점검하고 조치할 필요가 있다.

Apache 웹서버 관련 취약점에 대한 패치는 아래에서 받아 설치할 수 있다.

Official Patches for publically released versions of Apache :
http://www.apache.org/dist/httpd/patches/

다. 설정파일 및 데이터 백업

초기 서버 설정 파일들과 이후의 기본적인 설정파일들은 일반에 공개되거나 다른 변화가 일어나기 전에 백업해서 보관되어져야 한다. 또한 시스템 설정이 변경될 때마다 이력관리가 필요하고 다수의 수정이 있을 경우에는 반드시 백업을 하도록 한다.

설정파일 뿐 아니라 웹 데이터에 대한 백업도 정기적으로 이루어져야 한다. 웹서버의

중요도나 내용의 변화주기에 따라 달라질 수 있겠지만 가능하면 매일 백업을 받는 것이 안전하다. 또한 모든 백업 데이터는 정상적으로 동작될 수 있는지 검증되어져야만 하고, 백업 데이터 또한 서비스 중인 웹서버와 동등한 수준의 기밀성을 보장하여야만 한다.

이는 관리자의 실수에 의한 잘못된 설정이나 해킹으로 인해 웹서버가 정상적으로 동작하지 않을 경우 신속하게 복구하는데 많은 도움을 줄 수 있다.

CERTCC-KR에 접수된 사고 중에는 웹서버의 백업이 얼마나 중요한지를 느끼게 하는 사례가 있었다.

> 지방의 모 대학교에서 방학기간을 이용하여 웹서버 구축 및 홈페이지 작업을 했다. 방학이 거의 끝나갈 무렵 홈페이지 구축작업이 완성되어 웹서버를 인터넷에 연결시켰는데, 불과 며칠만에 웹서버가 해킹 당해 홈페이지 전체가 완전히 삭제되었다. 불행히도 이 대학교에서는 방학기간 중 서둘러 작업하느라 웹서버 백업이 전혀 이루어지지 않아 한 달여에 걸친 작업을 다시 해야만 했다.

6. 결론

지금까지 살펴본 바와 같이 Apache 웹서버에는 다양한 보안 설정 기능들이 있다. 하지만 많은 웹서버 관리자들은 이러한 기능들에 대해 알지 못하거나 시간이 없어서(?) 초기설정 상태의 웹서버를 인터넷에 연결하여 운영하는 경우가 대부분이다.

Apache 웹서버는 웹 컨텐츠만 만들어 넣으면 웹서비스를 할 수 있을 정도로 설치와 운영이 편리하지만, 그만큼 기본설정에서는 효율성만을 고려하여 불필요한 많은 기능들을 제공하고 있고, 이는 보안 허점으로 이어져 공격을 받을 수 있게 되는 경우가 많다.

웹서버는 인터넷을 통해 일반 대중에 공개되는 서버인 만큼 공격의 위험도 높고 실제 많은 피해가 보고되고 있으므로 안전한 웹서버 운용 노력이 요구된다.

[참고문헌]

Guidelines on Securing Public Web Servers
http:/ / csrc.nist.gov/ publications/ drafts/ PP-SecuringWebServers-RFC.pdf

Security Tips for Server Configuration
http:/ / httpd.apache.org/ docs/ misc/ security_tips.html

Apache Server Frequently Asked Questions
http:/ / httpd.apache.org/ docs/ misc/ FAQ.html

Apache Directives
http:/ / httpd.apache.org/ docs/ mod/ directives.html

Basic Apache Security Considerations
http:/ / rr.sans.org/ web/ apache_sec.php

The World Wide Web Security FAQ
http:/ / www.w3.org/ Security/ Faq/ wwwsf3.html#SVR-Q16

Apache Security
http:/ / www.apacheweek.com/ security/