Majd Hamdan

Libpng-1.2.5 bug

CSCI 1005
January 18, 2021

1 Bullet Points

- Download libpng-1.2.5.

We downloaded the libpng-1.2.5 from: https://sourceforge.net/projects/libpng/files/libpng12/older-releases/1.2.5/ where we selected the libpng-1.2.5.tar.gz file. Then we extracted the content of the file using:

```
[@majd 4]$ tar -xf "libpng-1.2.5.tar.gz"
```

- Configure it. In the resulting Makefile, add -fno-stack-protector to the CFLAGS variable.

To configure it, we ran ./configure. But, there was no configure file and alternate instructions were specified:

```
[@majd libpng-1.2.5]$ ./configure

  There is no "configure" script for Libpng-1.2.5.  Instead, please
  copy the appropriate makefile for your system from the "scripts"
  directory.  Read the INSTALL file for more details.
```

In "libpng-1.2.5/scripts/" we found the makefile for linux: makefile.linux. We moved to the libpng-1.2.5 folder, changed its name to Makefile, and added -fno-stack-protector to CFLAGS.

- Compile (but do not install!) it.

Finally we typed make in bash:

```
[@majd libpng-1.2.5]$ make
gcc -I../zlib -fno-stack-protector -Wall -O3 -funroll-loops     -c -o png.
    o png.c
gcc -I../zlib -fno-stack-protector -Wall -O3 -funroll-loops     -c -o
    pngset.o pngset.c
gcc -I../zlib -fno-stack-protector -Wall -O3 -funroll-loops     -c -o
    pngget.o pngget.c
gcc -I../zlib -fno-stack-p...
....
```

- In the contrib/gregbook subdirectory of the libpng source code:

  ```
  $ mv Makefile.unx Makefile
  ```

  Then edit the Makefile to include these variable definitions (you'll have to comment out any other definitions of these variables):

```
PNGINC = -I../..
PNGLIB = -L../.. -lpng12
ZINC = -I/usr/include
ZLIB = /usr/lib/libz.a
LDFLAGS = -lm
```

And finally compile the driver programs:

```
$ make rpng-x
```

```
[@majd libpng-1.2.5]$ cd contrib/gregbook/
[@majd gregbook]$ mv Makefile.unx Makefile
[majd gregbook]$ make rpng-x
gcc -c -O -Wall -I../.. -I/usr/include -I/usr/X11R6/include    rpng-x.c
gcc -c -O -Wall -I../.. -I/usr/include -I/usr/X11R6/include    readpng.c
gcc -lm -o rpng-x rpng-x.o readpng.o -L../.. -lpng12 /usr/lib/libz.a -L/
    usr/X11R6/lib -lX11 -lm
```

Before we ran make rpng-x, we had to comment out all the definitions for the variables PNGINC, PNGLIB,...and replace them with pete's definitions.

- Download this image: https://www.cs.middlebury.edu/~pjohnson/courses/w22-1005/lectures/06/bad-image.png, which will crash libpng-1.2.5.

Image is downloaded to lab 4 file.

- Elicit the crash by feeding the image to the rpng-x program. This is a bit non-trivial, because you need to tell rpng-x to use the version of libpng you just compiled rather than the system-installed version. You can do that thusly:

```
$ LD_LIBRARY_PATH=../../ ./rpng-x /path/to/image
```

We tried to open the img using the above command but we got a sig fault:

```
[@majd gregbook]$ LD_LIBRARY_PATH=../../  ./rpng-x /home/majd/Desktop/Labs
    /4/bad-image.png
libpng warning: Missing PLTE before tRNS
libpng warning: tRNS: CRC error
Segmentation fault (core dumped)
```

- Find the bug that causes the crash.

GDB is a great tool to find where the bug is. To run the program in gdb, we just add gdb before ./rpng-x:

```
[majd@majd gregbook]$ LD_LIBRARY_PATH=../../ gdb ./rpng-x /home/majd/
    Desktop/Labs/4/bad-image.png
GNU gdb (GDB) 11.1
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.
    html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./rpng-x...
(No debugging symbols found in ./rpng-x)
"/home/majd/Desktop/Labs/4/bad-image.png" is not a core dump: file format
    not recognized
```

We then provide the path to the image when we run the program per gdb note ”is not a core dump: file format not recognized”:

```
gdb) run /home/majd/Desktop/Labs/4/bad-image.png
Starting program: /home/majd/Desktop/Labs/4/libpng-1.2.5/contrib/gregbook/
    rpng-x /home/majd/Desktop/Labs/4/bad-image.png
libpng warning: Missing PLTE before tRNS
libpng warning: tRNS: CRC error

Program received signal SIGSEGV, Segmentation fault.
0x00007ffff7e3947b in png_handle_tRNS (png_ptr=<optimized out>, info_ptr=<
    optimized out>, length=<optimized out>) at pngrutil.c:1310
1310   }
```

We can see that the program crashed with a sig fault. A sig fault usually indicate that the program is trying to access a memory address that it doesn't have permission to access. Where is that happening? In the png_handle_rRNS function in the libpng-1.2.5/pngrutil.c file. When we open the file in atom, here is the content of the png_handle_rRNS function:

```
void /* PRIVATE */
png_handle_tRNS(png_structp png_ptr, png_infop info_ptr, png_uint_32
    length)
{
   png_byte readbuf[PNG_MAX_PALETTE_LENGTH];

   png_debug(1, "in png_handle_tRNS\n");

   if (!(png_ptr->mode & PNG_HAVE_IHDR))
```

```
      png_error(png_ptr, "Missing IHDR before tRNS");
   else if (png_ptr->mode & PNG_HAVE_IDAT)
   {
      png_warning(png_ptr, "Invalid tRNS after IDAT");
      png_crc_finish(png_ptr, length);
      return;
   }
   else if (info_ptr != NULL && (info_ptr->valid & PNG_INFO_tRNS))
   {
      png_warning(png_ptr, "Duplicate tRNS chunk");
      png_crc_finish(png_ptr, length);
      return;
   }

   if (png_ptr->color_type == PNG_COLOR_TYPE_PALETTE)
   {
      if (!(png_ptr->mode & PNG_HAVE_PLTE))
      {
         /* Should be an error, but we can cope with it */
         png_warning(png_ptr, "Missing PLTE before tRNS");
      }
      else if (length > (png_uint_32)png_ptr->num_palette)
      {
         png_warning(png_ptr, "Incorrect tRNS chunk length");
         png_crc_finish(png_ptr, length);
         return;
      }
      if (length == 0)
      {
         png_warning(png_ptr, "Zero length tRNS chunk");
         png_crc_finish(png_ptr, length);
         return;
      }

      png_crc_read(png_ptr, readbuf, (png_size_t)length);
      png_ptr->num_trans = (png_uint_16)length;
   }
   else if (png_ptr->color_type == PNG_COLOR_TYPE_RGB)
   {
      png_byte buf[6];

      if (length != 6)
      {
         png_warning(png_ptr, "Incorrect tRNS chunk length");
         png_crc_finish(png_ptr, length);
         return;
      }

      png_crc_read(png_ptr, buf, (png_size_t)length);
      png_ptr->num_trans = 1;
      png_ptr->trans_values.red = png_get_uint_16(buf);
      png_ptr->trans_values.green = png_get_uint_16(buf + 2);
      png_ptr->trans_values.blue = png_get_uint_16(buf + 4);
   }
   else if (png_ptr->color_type == PNG_COLOR_TYPE_GRAY)
   {
      png_byte buf[6];

      if (length != 2)
      {
         png_warning(png_ptr, "Incorrect tRNS chunk length");
         png_crc_finish(png_ptr, length);
         return;
      }

      png_crc_read(png_ptr, buf, 2);
      png_ptr->num_trans = 1;
      png_ptr->trans_values.gray = png_get_uint_16(buf);
   }
   else
   {
      png_warning(png_ptr, "tRNS chunk not allowed with alpha channel");
      png_crc_finish(png_ptr, length);
      return;
   }

   if (png_crc_finish(png_ptr, 0))
      return;

   png_set_tRNS(png_ptr, info_ptr, readbuf, png_ptr->num_trans,
      &(png_ptr->trans_values));
}
```

where is the error caused in this function? Lets put a break at this function and trace it step by step:

```
(gdb) b png_handle_tRNS
Breakpoint 1 at 0x7ffff7e39360: file pngrutil.c, line 1228.
(gdb) run /home/majd/Desktop/Labs/4/bad-image.png
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/majd/Desktop/Labs/4/libpng-1.2.5/contrib/gregbook/
    rpng-x /home/majd/Desktop/Labs/4/bad-image.png

Breakpoint 1, png_handle_tRNS (png_ptr=0x55555556b490, info_ptr=0
    x55555556f550, length=512) at pngrutil.c:1228
1228      if (!(png_ptr->mode & PNG_HAVE_IHDR))
```

After a couple of steps, we see that the error happened after line 1305:

```
Breakpoint 1, png_handle_tRNS (png_ptr=0x55555556b490, info_ptr=0
    x55555556f550, length=512) at pngrutil.c:1228
```

```
1228      if (!(png_ptr->mode & PNG_HAVE_IHDR))
(gdb) n
1230      else if (png_ptr->mode & PNG_HAVE_IDAT)
(gdb)
1236      else if (info_ptr != NULL && (info_ptr->valid & PNG_INFO_tRNS))
(gdb)
1243      if (png_ptr->color_type == PNG_COLOR_TYPE_PALETTE)
(gdb)
1245         if (!(png_ptr->mode & PNG_HAVE_PLTE))
(gdb)
1248            png_warning(png_ptr, "Missing PLTE before tRNS");
(gdb)
libpng warning: Missing PLTE before tRNS
1256         if (length == 0)
(gdb)
1263         png_crc_read(png_ptr, readbuf, (png_size_t)length);
(gdb)
1264         png_ptr->num_trans = (png_uint_16)length;
(gdb)
1305      if (png_crc_finish(png_ptr, 0))
(gdb)
libpng warning: tRNS: CRC error

Program received signal SIGSEGV, Segmentation fault.
0x00007ffff7e3947b in png_handle_tRNS (png_ptr=<optimized out>, info_ptr=<
    optimized out>, length=<optimized out>) at pngrutil.c:1310
1310   }
```

Lets now disassemble and find exactly at which assembly command it sig fault:

```
(gdb) disas
Dump of assembler code for function png_handle_tRNS:
   0x00007ffff7e39360 <+0>: push   %r14
   0x00007ffff7e39362 <+2>: push   %r13
   0x00007ffff7e39364 <+4>: mov    %rsi,%r13
   0x00007ffff7e39367 <+7>: push   %r12
   0x00007ffff7e39369 <+9>: mov    %rdx,%r12
.
.
.
   0x00007ffff7e39477 <+279>: pop    %r13
   0x00007ffff7e39479 <+281>: pop    %r14
=> 0x00007ffff7e3947b <+283>: ret
   0x00007ffff7e3947c <+284>: nopl   0x0(%rax)
   0x00007ffff7e39480 <+288>: cmp    $0x6
.
.
.
```

Great! We see that it sigfaulted at the return command which causes rip to change its value to address of the instruction we must return to after the implementation of png_handle_rRNS is done. This indicate that the return address of this function is corrupted. How? buffer overflow. A buffer in the png_handle_rRNS had more data written into it than it can store (that its size), so the stack was corrupted with info changing the return address of this function into an inaccessible address.

What buffer is that? and where did this exactly happen? There is only one buffer in png_handle_rRNS function and it is readbuf:

```
.
.
.
   png_byte readbuf[PNG_MAX_PALETTE_LENGTH];
.
.
.
```

and there are only two places that this buffer is being written to, on line 1263 and line 1308:

```
.
.
.
1263: png_crc_read(png_ptr, readbuf, (png_size_t)length);
.
.
.
1308: png_set_tRNS(png_ptr, info_ptr, readbuf, png_ptr->num_trans,
      &(png_ptr->trans_values));
.
.
.
```

When we stepped through the png_handle_rRNS function before, we saw that the sigfault happens at line 1305 which is called right after line 1263. So the call to png_crc_read(png_ptr, readbuf, (png_size_t)length); at line 1263 is the likely cause for the error. Something in the function png_crc_read() if overflowing readbuf. When searching for this function in pngrutil.c, we find that it is defined as:

```
/* Read data, and (optionally) run it through the CRC. */
void /* PRIVATE */
png_crc_read(png_structp png_ptr, png_bytep buf, png_size_t length)
{
   png_read_data(png_ptr, buf, length);
   png_calculate_crc(png_ptr, buf, length);
}
```

From reading the description to this function and drawing similarities to the "read" system call, we understand that this function is reading "length" bytes into readbuf from png-ptr. Since it is overflowing readbuf, then "length" must be larger