

What I first did was compiled AFL:

```
(fyaish@kali)-[~/cse5472/lab1/AFL-2.57b]
$ make
[*] Checking for the ability to compile x86 code...
[+] Everything seems to be working, ready to compile.
cc -O3 -funroll-loops -Wall -D_FORTIFY_SOURCE=2 -g -Wno-pointer-sign -DAFL_PATH="/usr/local/lib/afl/" -DDOC_PATH="/usr/local/share/doc/afl/" -DBIN_PATH="/usr/local/bin/" afl-gcc.c -o afl-gcc -ldl
set -e; for i in afl-g++ afl-clang afl-clang++; do ln -sf afl-gcc $i; done
cc -O3 -funroll-loops -Wall -D_FORTIFY_SOURCE=2 -g -Wno-pointer-sign -DAFL_PATH="/usr/local/lib/afl/" -DDOC_PATH="/usr/local/share/doc/afl/" -DBIN_PATH="/usr/local/bin/" afl-fuzz.c -o afl-fuzz -ldl
cc -O3 -funroll-loops -Wall -D_FORTIFY_SOURCE=2 -g -Wno-pointer-sign -DAFL_PATH="/usr/local/lib/afl/" -DDOC_PATH="/usr/local/share/doc/afl/" -DBIN_PATH="/usr/local/bin/" afl-showmap.c -o afl-showmap -ldl
cc -O3 -funroll-loops -Wall -D_FORTIFY_SOURCE=2 -g -Wno-pointer-sign -DAFL_PATH="/usr/local/lib/afl/" -DDOC_PATH="/usr/local/share/doc/afl/" -DBIN_PATH="/usr/local/bin/" afl-tmin.c -o afl-tmin -ldl
cc -O3 -funroll-loops -Wall -D_FORTIFY_SOURCE=2 -g -Wno-pointer-sign -DAFL_PATH="/usr/local/lib/afl/" -DDOC_PATH="/usr/local/share/doc/afl/" -DBIN_PATH="/usr/local/bin/" afl-gotcpu.c -o afl-gotcpu -ldl
cc -O3 -funroll-loops -Wall -D_FORTIFY_SOURCE=2 -g -Wno-pointer-sign -DAFL_PATH="/usr/local/lib/afl/" -DDOC_PATH="/usr/local/share/doc/afl/" -DBIN_PATH="/usr/local/bin/" afl-analyze.c -o afl-analyze -ldl
cc -O3 -funroll-loops -Wall -D_FORTIFY_SOURCE=2 -g -Wno-pointer-sign -DAFL_PATH="/usr/local/lib/afl/" -DDOC_PATH="/usr/local/share/doc/afl/" -DBIN_PATH="/usr/local/bin/" afl-as.c -o afl-as -ldl
ln -sf afl-as as
[*] Testing the CC wrapper and instrumentation output...
unset AFL_USE_ASAN AFL_USE_MSAN; AFL_QUIET=1 AFL_INST_RATIO=100 AFL_PATH=. ./afl-gcc -O3 -funroll-loops -Wall -D_FORTIFY_SOURCE=2 -g -Wno-pointer-sign -DAFL_PATH="/usr/local/lib/afl/" -DDOC_PATH="/usr/local/share/doc/afl/" -DBIN_PATH="/usr/local/bin/" test-instr.c -o test-instr -ldl
./afl-showmap -m none -q -o .test-instr0 ./test-instr < /dev/null
echo 1 | ./afl-showmap -m none -q -o .test-instr1 ./test-instr
[+] All right, the instrumentation seems to be working!
[+] LLVM users: see llvm_mode/README.llvm for a faster alternative to afl-gcc.
[+] All done! Be sure to review README - it's pretty short and useful.
```

Then I went into libpng-1.2.5 Makefile and changed the C compiler to afl-gcc and the prefix to my lab1 directory:

```
CC=/home/fyaish/cse5472/lab1/AFL-2.57b/afl-gcc

# where "make install" puts libpng12.a, libpng12.so*,
# libpng12/png.h and libpng12/pngconf.h
# Prefix must be a full pathname.
prefix=/home/fyaish/cse5472/lab1
```

Next I compiled libpng-1.2.5 by using the make and make install command in the libpng-1.2.5 directory:

```
(fyaieish@kali)-[~/cse5472/lab1/libpng-1.2.5]
$ make
/home/fyaieish/cse5472/lab1/AFL-2.57b/afl-gcc -I../zlib -Wall -O3 -funroll-loops -c -o png.o png.c
afl-cc 2.57b by <lcamtuf@google.com>
afl-as 2.57b by <lcamtuf@google.com>
[+] Instrumented 158 locations (64-bit, non-hardened mode, ratio 100%).
/home/fyaieish/cse5472/lab1/AFL-2.57b/afl-gcc -I../zlib -Wall -O3 -funroll-loops -c -o pngset.o pngset.c
afl-cc 2.57b by <lcamtuf@google.com>
afl-as 2.57b by <lcamtuf@google.com>
[+] Instrumented 253 locations (64-bit, non-hardened mode, ratio 100%).
/home/fyaieish/cse5472/lab1/AFL-2.57b/afl-gcc -I../zlib -Wall -O3 -funroll-loops -c -o pngget.o pngget.c
afl-cc 2.57b by <lcamtuf@google.com>
afl-as 2.57b by <lcamtuf@google.com>
[+] Instrumented 265 locations (64-bit, non-hardened mode, ratio 100%).
/home/fyaieish/cse5472/lab1/AFL-2.57b/afl-gcc -I../zlib -Wall -O3 -funroll-loops -c -o pngutil.o pngutil.c
afl-cc 2.57b by <lcamtuf@google.com>
afl-as 2.57b by <lcamtuf@google.com>
pngutil.c: In function 'png_decompress_chunk.part.0':
pngutil.c:273:70: warning: 'chunk' directive writing 6 bytes into a region of size between 5 and 9 [-Wformat-overflow=]
    273 |         sprintf(msg,"Buffer error in compressed datastream in %s chunk",
        |         ~~~~~~
        |         ^~~~~~
pngutil.c:273:13: note: 'sprintf' output between 48 and 52 bytes into a destination of size 50
    273 |         sprintf(msg,"Buffer error in compressed datastream in %s chunk",
        |         ^~~~~~
    274 |         png_ptr->chunk_name);
afl-as 2.57b by <lcamtuf@google.com>
[+] Instrumented 1238 locations (64-bit, non-hardened mode, ratio 100%).
/home/fyaieish/cse5472/lab1/AFL-2.57b/afl-gcc -I../zlib -Wall -O3 -funroll-loops -c -o pngtrans.o pngtrans.c
afl-cc 2.57b by <lcamtuf@google.com>
afl-as 2.57b by <lcamtuf@google.com>
[+] Instrumented 482 locations (64-bit, non-hardened mode, ratio 100%).
/home/fyaieish/cse5472/lab1/AFL-2.57b/afl-gcc -I../zlib -Wall -O3 -funroll-loops -c -o pngwutil.o pngwutil.c
afl-cc 2.57b by <lcamtuf@google.com>
pngwutil.c: In function 'png_write_compressed_data_out':
pngwutil.c:348:4: warning: this 'if' clause does not guard... [-Wmisleading-indentation]
    348 |     if (comp->max_output_ptr != 0)

(fyaieish@kali)-[~/cse5472/lab1/libpng-1.2.5]
$ make install
cp png.h pngconf.h /home/fyaieish/cse5472/lab1/include/libpng12
chmod 644 /home/fyaieish/cse5472/lab1/include/libpng12/png.h /home/fyaieish/cse5472/lab1/include/libpng12/pngconf.h
(cd /home/fyaieish/cse5472/lab1/include; ln -sf libpng12 libpng; ln -sf libpng12/* .)
cp libpng.a /home/fyaieish/cse5472/lab1/lib/libpng12.a
chmod 644 /home/fyaieish/cse5472/lab1/lib/libpng12.a
(cd /home/fyaieish/cse5472/lab1/lib; ln -sf libpng12.a libpng.a)
/home/fyaieish/cse5472/lab1/AFL-2.57b/afl-gcc -shared -WL,-soname,libpng.so.3 \
-o libpng.so.3.1.2.5 \
png.pic.o pngset.pic.o pngget.pic.o pngutil.pic.o pngtrans.pic.o pngwutil.pic.o pngread.pic.o pngrio.pic.o pngwio.p
ic.o pngwrite.pic.o pngtran.pic.o pngwtran.pic.o pngmem.pic.o pngerror.pic.o pngpread.pic.o
afl-cc 2.57b by <lcamtuf@google.com>
cp libpng12.so.0.1.2.5 /home/fyaieish/cse5472/lab1/lib
cp libpng.so.3.1.2.5 /home/fyaieish/cse5472/lab1/lib
chmod 755 /home/fyaieish/cse5472/lab1/lib/libpng12.so.0.1.2.5
chmod 755 /home/fyaieish/cse5472/lab1/lib/libpng.so.3.1.2.5
(cd /home/fyaieish/cse5472/lab1/lib; \
ln -sf libpng.so.3.1.2.5 libpng.so.3; \
ln -sf libpng.so.3 libpng.so; \
ln -sf libpng12.so.0.1.2.5 libpng12.so.0; \
ln -sf libpng12.so.0 libpng12.so)
cp libpng.pc /home/fyaieish/cse5472/lab1/lib/pkgconfig/libpng12.pc
chmod 644 /home/fyaieish/cse5472/lab1/lib/pkgconfig/libpng12.pc
(cd /home/fyaieish/cse5472/lab1/lib/pkgconfig; ln -sf libpng12.pc libpng.pc)
cp libpng.3 /home/fyaieish/cse5472/lab1/man/man3
cp libpngpf.3 /home/fyaieish/cse5472/lab1/man/man3
cp png.5 /home/fyaieish/cse5472/lab1/man/man5
cp libpng-config /home/fyaieish/cse5472/lab1/bin/libpng12-config
chmod 755 /home/fyaieish/cse5472/lab1/bin/libpng12-config
(cd /home/fyaieish/cse5472/lab1/bin; ln -sf libpng12-config libpng-config)
```

Here is where I compile pngslap using afl-gcc by setting the path to afl-gcc and adding the path to the include directory and lib directory since I changed the prefix earlier in the libpng-1.2.5 Makefile:

```
(fyaish@kali)~[cse5472/lab1]
$ /home/fyaish/cse5472/lab1/AFL-2.57b/afl-gcc -g -o pngslap pngslap.c -I /home/fyaish/cse5472/lab1/include -L /home/fyaish/cse5472/lab1/lib -lpng -lz -lm
afl-cc 2.57b by <lcamtuf@google.com>
afl-as 2.57b by <lcamtuf@google.com>
[+] Instrumented 12 locations (64-bit, non-hardened mode, ratio 100%).
```

Here is the command used to run afl-fuzz and get multiple crashes:

```
(fyaish@kali)~[cse5472/lab1]
$ LD_LIBRARY_PATH=/home/fyaish/cse5472/lab1/lib ./AFL-2.57b/afl-fuzz -i inputs -o outputs -- ./pngslap @ /dev/null
```

```
american fuzzy lop 2.57b (pngslap)

process timing
  run time : 0 days, 0 hrs, 10 min, 36 sec
  last new path : 0 days, 0 hrs, 0 min, 34 sec
  last uniq crash : 0 days, 0 hrs, 1 min, 5 sec
  last uniq hang : none seen yet
cycle progress
  now processing : 84 (97.67%)
  paths timed out : 0 (0.00%)
stage progress
  now trying : bitflip 1/1
  stage execs : 54.9k/131k (41.72%)
  total execs : 1.08M
  exec speed : 1594/sec
fuzzing strategy yields
  bit flips : 38/121k, 1/121k, 2/121k
  byte flips : 0/15.2k, 0/2957, 0/2895
  arithmetics : 8/166k, 0/96.7k, 0/48.4k
  known ints : 0/13.9k, 9/62.5k, 2/106k
  dictionary : 0/0, 0/0, 17/29.6k
  havoc : 13/108k, 0/0
  trim : 40.32%/8502, 79.99%

overall results
  cycles done : 2
  total paths : 86
  uniq crashes : 6
  uniq hangs : 0

map coverage
  map density : 0.38% / 0.61%
  count coverage : 1.90 bits/tuple
findings in depth
  favored paths : 28 (32.56%)
  new edges on : 37 (43.02%)
  total crashes : 18 (6 unique)
  total tmouts : 0 (0 unique)
path geometry
  levels : 6
  pending : 36
  pend fav : 1
  own finds : 85
  imported : n/a
  stability : 100.00%

[cpu000: 50%]
```

Once afl-fuzz found crashes I stopped running afl-fuzz and went to the outputs/crashes directory to see what triggered the crashes. I then used one of these as my PoC(proof of compromise) to run gdb and find the vulnerability. After running gdb and doing a back trace it led me to a function called png_handle_tRNS inside the file pngutil.c.

```
(fyaish@kali)~[cse5472/lab1]
$ LD_LIBRARY_PATH=/home/fyaish/cse5472/lab1/lib gdb --args ./pngslap /home/fyaish/cse5472/lab1/outputs/crashes/id:000000,sig:11,src:000000,op:arith8,pos:68,val:+16 /dev/null
GNU gdb (Debian 13.2-1) 13.2
Copyright (C) 2023 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-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 ./pngslap ...
(gdb) r
Starting program: /home/fyaish/cse5472/lab1/pngslap /home/fyaish/cse5472/lab1/outputs/crashes/id:000000,sig:11,src:000000,op:arith8,pos:68,val:+16 /dev/null
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
libpng warning: Missing PLTE before tRNS
libpng warning: tRNS: CRC error

Program received signal SIGSEGV, Segmentation fault.
0x00007ffff7f1a5e1 in png_handle_tRNS (png_ptr=<optimized out>, info_ptr=<optimized out>, length=<optimized out>) at pngutil.c:1304
warning: Source file is more recent than executable.
1304         6(png_ptr->trans_values));
(gdb) bt
#0  0x00007ffff7f1a5e1 in png_handle_tRNS (png_ptr=<optimized out>, info_ptr=<optimized out>, length=<optimized out>) at pngutil.c:1304
#1  0x133a3534c6c8c651 in ?? ()
#2  0x61745c506a871313 in ?? ()
#3  0x84810d0d0d485764 in ?? ()
#4  0x0689847e8685858c in ?? ()
#5  0xb6b8b60705070607 in ?? ()
#6  0x0304605c5b050505 in ?? ()
#7  0x02b3b4b3245e5e04 in ?? ()
#8  0x0001000101010201 in ?? ()
#9  0xaeacfa1616aeaeae in ?? ()
#10 0x39727371dc8e8cac in ?? ()
#11 0x173522212b2c3b3a in ?? ()
#12 0x1f2535323391aca5 in ?? ()
#13 0xeb93949345acac1a in ?? ()
```

Once afl-fuzz found crashes I stopped running afl-fuzz and went to the outputs/crashes directory to see what triggered the crashes. I then used one of these as my PoC (proof of compromise) to run gdb and find the vulnerability. After running gdb and adding a bit it lead me to the function called

Then I did more debugging and found the specific location for the vulnerability which was in lines 1240-1258 in the function called png_handle_tRNS inside the file pngutil.c:

```

1238 if (png_ptr->color_type == PNG_COLOR_TYPE_PALETTE)
1239 {
1240     if (!(png_ptr->mode & PNG_HAVE_PLTE))
1241     {
1242         /* Should be an error, but we can cope with it */
1243         png_warning(png_ptr, "Missing PLTE before tRNS");
1244     }
1245     else if (length > (png_uint_32)png_ptr->num_palette)
1246     {
1247         png_warning(png_ptr, "Incorrect tRNS chunk length");
1248         png_crc_finish(png_ptr, length);
1249         return;
1250     }
1251     if (length == 0)
1252     {
1253         png_warning(png_ptr, "Zero length tRNS chunk");
1254         png_crc_finish(png_ptr, length);
1255         return;
1256     }
1257
1258     png_crc_read(png_ptr, readbuf, (png_size_t)length);

```

The code checks the size of length which is used later in the png_crc_read. Before it checks the length it checks png_ptr->mode in the first if statement. If the first if statement is true then the else if will be skipped and there will be no check on the length. If there is no check on length it will still be called in png_crc_read possibly leading to a stack buffer overflow. The CWE class for this vulnerability is CWE-121(Stack-based buffer overflow). The condition required to trigger it is when the first if statement is executed and the else if length check is skipped.

To fix the vulnerability, we need to ensure that length is always checked; to do this, we change the else if to an if statement. The fix for the vulnerability:

```
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");
}
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);
```