What I first did was compiled AFL:

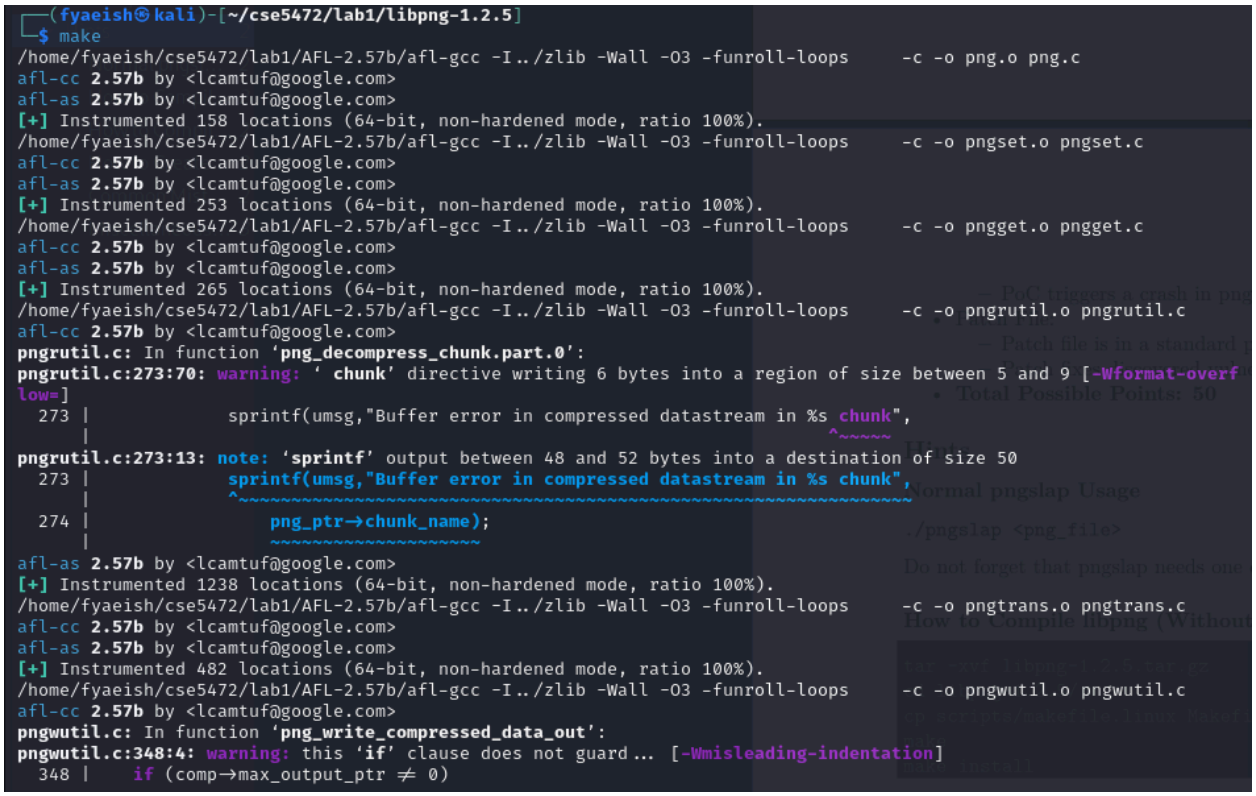
```
┌──(fyaeish㉿kali)-[~/cse5472/lab1/AFL-2.57b]
└─$ make
[*] Checking for the ability to compile x86 code...
[+] Everything seems to be working, ready to compile.
cc -O3 -funroll-loops -Wall -D_FORTIFY_SOURCE=2 -g -Wno-pointer-sign -DAFL_PATH=\"/us
r/local/lib/afl\" -DDOC_PATH=\"/usr/local/share/doc/afl\" -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=\"/us
r/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=\"/us
r/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=\"/us
r/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=\"/us
r/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=\"/us
r/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=\"/us
r/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/l
ocal/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/fyaeish/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/fyaeish/cse5472/lab1
```

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



```
┌──(fyaeish㉿kali)-[~/cse5472/lab1/libpng-1.2.5]
└─$ make
/home/fyaeish/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/fyaeish/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/fyaeish/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/fyaeish/cse5472/lab1/AFL-2.57b/afl-gcc -I../zlib -Wall -O3 -funroll-loops     -c -o pngrutil.o pngrutil.c
afl-cc 2.57b by <lcamtuf@google.com>
pngrutil.c: In function 'png_decompress_chunk.part.0':
pngrutil.c:273:70: warning: ' chunk' directive writing 6 bytes into a region of size between 5 and 9 [-Wformat-overf
low=]
  273 |             sprintf(umsg,"Buffer error in compressed datastream in %s chunk",
      |                                                                      ^~~~~~
pngrutil.c:273:13: note: 'sprintf' output between 48 and 52 bytes into a destination of size 50
  273 |             sprintf(umsg,"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/fyaeish/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/fyaeish/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)
```

```
┌──(fyaeish㉿kali)-[~/cse5472/lab1/libpng-1.2.5]
└─$ make install
cp png.h pngconf.h /home/fyaeish/cse5472/lab1/include/libpng12
chmod 644 /home/fyaeish/cse5472/lab1/include/libpng12/png.h /home/fyaeish/cse5472/lab1/include/libpng12/pngconf.h
(cd /home/fyaeish/cse5472/lab1/include; ln -sf libpng12 libpng; ln -sf libpng12/* .)
cp libpng.a /home/fyaeish/cse5472/lab1/lib/libpng12.a
chmod 644 /home/fyaeish/cse5472/lab1/lib/libpng12.a
(cd /home/fyaeish/cse5472/lab1/lib; ln -sf libpng12.a libpng.a)
/home/fyaeish/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 pngrutil.pic.o pngtrans.pic.o pngwutil.pic.o pngread.pic.o pngrio.pic.o pngwio.p
ic.o pngwrite.pic.o pngrtran.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/fyaeish/cse5472/lab1/lib
cp libpng.so.3.1.2.5 /home/fyaeish/cse5472/lab1/lib
chmod 755 /home/fyaeish/cse5472/lab1/lib/libpng12.so.0.1.2.5
chmod 755 /home/fyaeish/cse5472/lab1/lib/libpng.so.3.1.2.5
(cd /home/fyaeish/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/fyaeish/cse5472/lab1/lib/pkgconfig/libpng12.pc
chmod 644 /home/fyaeish/cse5472/lab1/lib/pkgconfig/libpng12.pc
(cd /home/fyaeish/cse5472/lab1/lib/pkgconfig; ln -sf libpng12.pc libpng.pc)
cp libpng.3 /home/fyaeish/cse5472/lab1/man/man3
cp libpngpf.3 /home/fyaeish/cse5472/lab1/man/man3
cp png.5 /home/fyaeish/cse5472/lab1/man/man5
cp libpng-config /home/fyaeish/cse5472/lab1/bin/libpng12-config
chmod 755 /home/fyaeish/cse5472/lab1/bin/libpng12-config
(cd /home/fyaeish/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:

```
┌──(fyaeish㉿kali)-[~/cse5472/lab1]
└─$ /home/fyaeish/cse5472/lab1/AFL-2.57b/afl-gcc -g -o pngslap pngslap.c -I /home/fyaeish/cse5472/lab1/include -L /h
ome/fyaeish/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%).
```

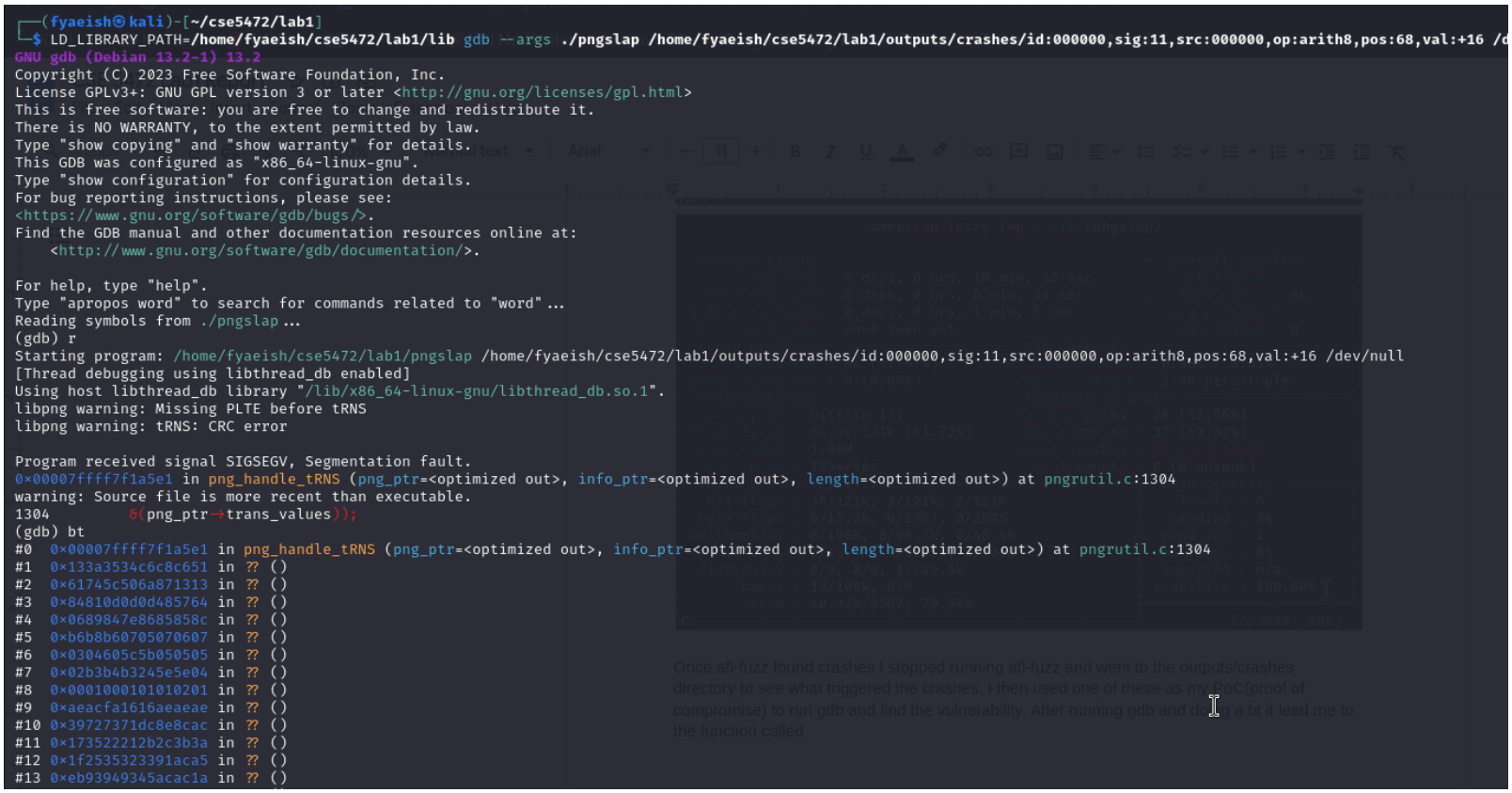
Here is the command used to run afl-fuzz and get multiple crashes:

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

```
                    american fuzzy lop 2.57b (pngslap)

┌─ process timing ─────────────────────┬─ overall results ─────┐
│        run time : 0 days, 0 hrs, 10 min, 36 sec │  cycles done : 2      │
│   last new path : 0 days, 0 hrs, 0 min, 34 sec  │  total paths : 86     │
│ last uniq crash : 0 days, 0 hrs, 1 min, 5 sec   │ uniq crashes : 6      │
│  last uniq hang : none seen yet                 │   uniq hangs : 0      │
├─ cycle progress ─────┬─ map coverage ──────────┴──────────────┤
│  now processing : 84 (97.67%)  │   map density : 0.38% / 0.61%                │
│ paths timed out : 0 (0.00%)    │ count coverage : 1.90 bits/tuple             │
├─ stage progress ─────┼─ findings in depth ─────┬─ path geometry ──┤
│  now trying : bitflip 1/1      │ favored paths : 28 (32.56%)  │   levels : 6     │
│ stage execs : 54.9k/131k (41.72%) │  new edges on : 37 (43.02%)  │  pending : 36    │
│ total execs : 1.08M            │ total crashes : 18 (6 unique)│ pend fav : 1     │
│  exec speed : 1594/sec         │  total tmouts : 0 (0 unique) │ own finds : 85   │
├─ fuzzing strategy yields ─────┴──────────────┴──  imported : n/a   │
│   bit flips : 38/121k, 1/121k, 2/121k                           │ stability : 100.00% │
│  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%                               │
^C                                                                 [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 pngrutil.c.



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 pngrutil.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);
```