CSE 5472: Fuzzing Lab

Objective

Learn about the merits and challenges of fuzz testing by finding, documenting, and patching a real vulnerability in an open source project.

Deliverable

1. Written report (PDF format, contents described below)
2. Vulnerability triggering input, a.k.a proof-of-compromise (as generated by AFL)
3. Patch file (generated using `diff`, details below)

Environment

Please use a Linux environment (Debian or Ubuntu preferred). Virtual machines are fine.

If you plan to use `stdlinux`, you will need to set the environment variable `AFL_I_DONT_CARE_ABOUT_MISSING_CRASHES=1` before running `afl-fuzz`, otherwise AFL will print an error about core dumps and refuse to start. See this document for more details.

Provided Materials

- `libpng-1.2.5.tar.gz`: A vulnerable version of libpng, a popular library for reading PNG image files.
- `pngslap.c`: An example program that uses libpng to read a PNG file.
- `seed.png`: A seed PNG file to bootstrap your fuzzing campaign.

Recommended Tools

- AFL
- gcc
- gdb
- diff

Note: In most Linux environments, `gcc`, `gdb`, and `diff` can be installed using APT: `sudo apt install build-essential gdb diff`

Tasks

1. Fuzz `pngslap` using AFL and find a crashing input.
2. Document the vulnerability you found.
3. Rewrite the code in libpng to patch the vulnerability.
4. Recompile libpng and verify that your patch fixes the vulnerability.

Grading

- PDF Report:
  - Report explains how libpng and pngslap were compiled with AFL instrumentation. Explanation includes terminal commands and screenshots (required!). **10 points**
  - Report provides terminal command used to run AFL. **1 points**
  - Report identifies location (filename and line numbers) in the libpng source code that contains the vulnerability. **10 points**
  - Report describes the vulnerability, including the conditions required to trigger it. **10 points**
  - Report provides the CWE class of the vulnerability (e.g., CWE-121). **2 points**
  - Report describes how the vulnerability can be fixed. **5 points**
- Proof-of-Compromise:

  - PoC triggers a crash in pngslap. **5 points**
- Patch File:
  - Patch file is in a standard patch format, as generated by `diff`. **2 points**
  - Patch fixes discovered vulnerability. **5 points**
- **Total Possible Points: 50**

Hints

Normal pngslap Usage

`./pngslap <png_file>`

Do not forget that pngslap needs one command line argument, a PNG file path!

How to Compile libpng (Without AFL)

```
tar -xvf libpng-1.2.5.tar.gz
cd libpng-1.2.5/
cp scripts/makefile.linux Makefile
make
make install
```

Note: You will need to modify `Makefile` to invoke AFL's `afl-gcc` instead of `gcc`.

Note: It is recommended that you also change `prefix` in `Makefile` to an empty local directory so you do not install a vulnerable version of libpng into your global system paths!

How to Compile pngslap (Without AFL)

After compiling and installing libpng (see above):

```
gcc -g -o pngslap pngslap.c -lpng -lz -lm
```

Note: To use pngslap with AFL, you will need to use `afl-gcc` instead of `gcc`.

Note: If you changed `prefix` in `Makefile`, the compile command is:

`gcc -g -o pngslap pngslap.c -I </path/to/install/include> -L </path/to/install/lib> -lpng -lz -lm`.

Note: If you used `-L` when compiling, you need to use

`LD_LIBRARY_PATH` to run pngslap:

`LD_LIBRARY_PATH=/path/to/install/lib ./pngslap <png_file>`.

How to Create a Diff Patch

```
diff -u file.c.old file.c > fix.patch
```

Common Mistakes

`./pngslap: error while loading shared libraries: libpng.so.3: cannot open shared object file: No such file or directory`

Do you need `LD_LIBRARY_PATH`?

AFL: `PROGRAM ABORT : No instrumentation detected`

Did you compile libpng *and* pngslap using `afl-gcc`?

AFL Interface: `(odd, check syntax!)`

Look at `overall results` in the AFL interface. Is `cycles done` greater than 1 and `total paths` less than 50? If so, pngslap is likely not receiving any command line arguments (it needs 1; a PNG file path).

It is also possible that you instrumented pngslap, but forgot libpng.

Miscellaneous Hints

- AFL should be able to find a crash within 1 minute if you use the provided seed.
- Make sure libpng is compiled with debug symbols. This will make tracing the bug back to source code easier. You can check using `file`:

```
$ file install/lib/libpng.so.3.1.2.5
install/lib/libpng.so.3.1.2.5: ELF 64-bit LSB shared object, x86-64, version 1
(SYSV), dynamically linked, with debug_info, not stripped
```