Fuzzing With AFL-Fuzz, a Practical Example (AFL vs Binutils)

2015-04-30

It's been a few weeks I've been playing with afl-fuzz (*american fuzzy lop*), a great tool from lcamtuf which uses binary instrumentation to create edge-cases for a given software, the description on the website is:

```
American fuzzy lop is a security-oriented fuzzer that employs a novel type
of compile-time
instrumentation and genetic algorithms to automatically discover clean,
interesting test cases
that trigger new internal states in the targeted binary.
This substantially improves the functional coverage for the fuzzed code.
The compact synthesized
corpora produced by the tool are also useful for seeding other, more
labor- or resource-intensive
testing regimes down the road.
```

Ok nice ... but what does this actually mean?

Binary Instrumentation

The first component of AFL is a wrapper for GCC/CLang compilers that will inject during compilation its own assembly code into the target software.
The fuzzer will use this code to trace execution paths while feeding the system with new inputs and to determine if a new mutation of the input is able to trigger known or unkown (new) execution paths.

If you can't/don't want to recompile the source code of the target program, AFL also supports a QEMU mode, an extension that leverages the QEMU "user emulation" mode and allows callers to obtain instrumentation output for black-box, closed-source binaries. This mechanism can be then used by

afl-fuzz to stress-test targets that couldn't be built with afl-gcc.

Installation

The installation of AFL is trivial, just download the latest version of the source code, extract it and do the usual:

```
make
sudo make install
```

A practical example

Being AFL particularly well suited for programs that accept a file as input, I thought about trying it against the **binutils** and specifically against the **readelf** binary ... AFL found eight distinct crashes cases (potentially exploitable) in the first five minutes of elaboration, and more than 30 after less than one hour!

First of all, you need to download a copy of the target program source code and extract it, once you're inside the extracted folder, you will need to override the **CC** (or **CXX** if it's a C++ software being compiled with g++ instead of gcc) environment variable before triggering the **configure** script and finally proceed with the compilation as usual.

```
cd ~/binutils-2.25
CC=afl-gcc ./configure
make
```

NOTE: If you want to use clang instead of gcc, you need to set CC to **afl-clang**.

During the compilation, you will notice a few messages from AFL informing you that it's correctly injecting its instrumentations.

Once finished, you want to tune up your configuration, the following command will instruct the system to output coredumps as files instead of

sending them to a specific crash handler app.

```
# echo core > /proc/sys/kernel/core_pattern
```

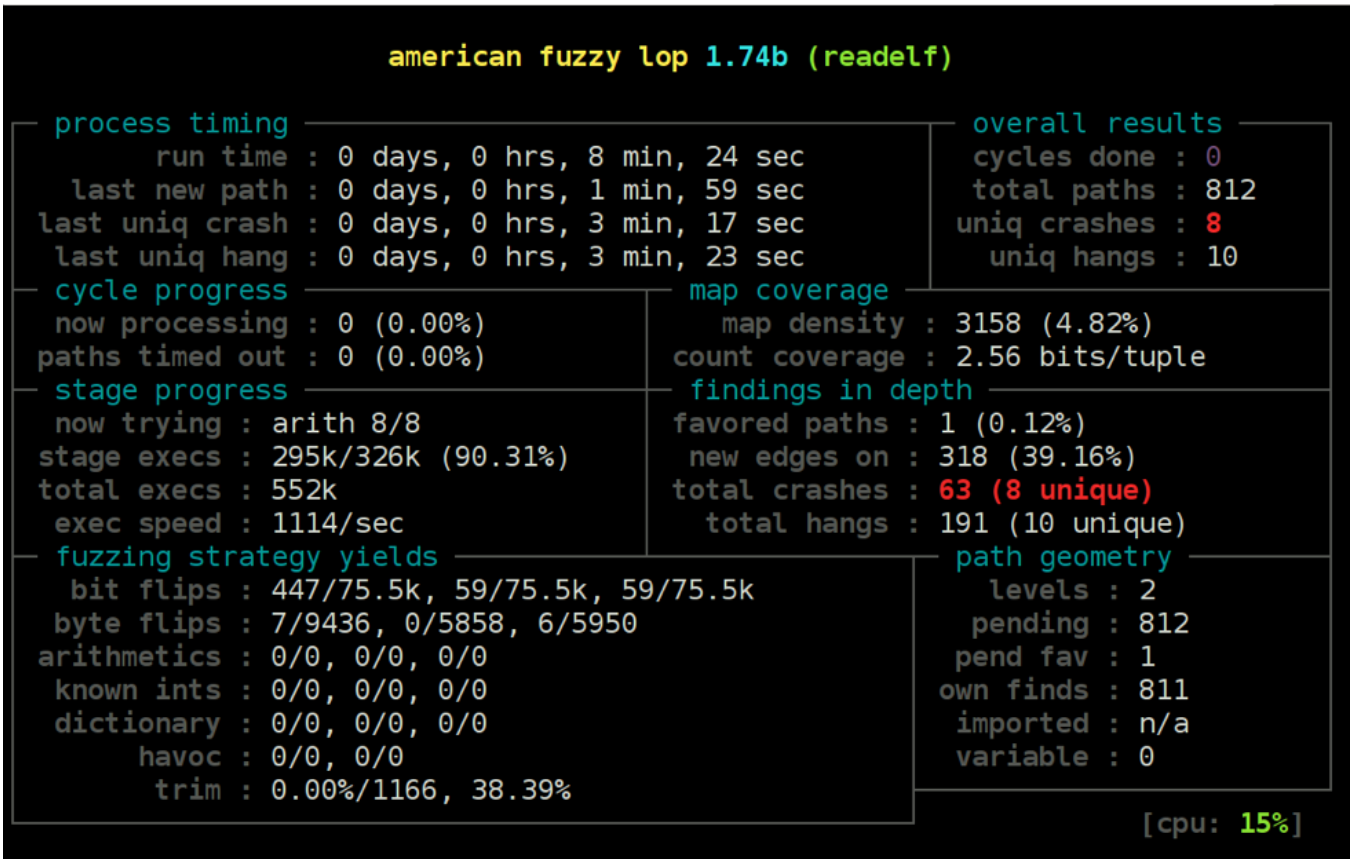
Now, we need to create an **input** folder where we will put our legit ELF file, in our case the **/bin/ps** command, (AFL will use it as a base template) and an **output** folder where the fuzzer will store its state but more important all the generated samples that make the application crash or hang.

```
cd ~/binutils-2.25
mkdir afl_in afl_out
cp /bin/ps afl_in/
```

And finally, let's run the fuzzer:

```
cd ~/binutils-2.25
afl-fuzz -i afl_in -o afl_out ./binutils/readelf -a @@
```

Eventually, we will start to see something like the following:



As you can see, the red number **8** on the top right is the total number of **unique** crashes the system was able to trigger so far.
In the **afl_out/crashes** folder you will find the files that made readelf crash, for instance if I try to execute readelf against the first crashing sample (inside GDB), I get the following:

```
*** buffer overflow detected ***: /home/evilsocket/binutils-
2.25/binutils/readelf terminated
======= Backtrace: =========
/lib/x86_64-linux-gnu/libc.so.6(+0x78c4e)[0x7ffff786cc4e]
/lib/x86_64-linux-gnu/libc.so.6(__fortify_fail+0x5c)[0x7ffff790ce8c]
/lib/x86_64-linux-gnu/libc.so.6(+0x116e80)[0x7ffff790ae80]
/lib/x86_64-linux-gnu/libc.so.6(__strcat_chk+0x5d)[0x7ffff790a05d]
/home/evilsocket/binutils-2.25/binutils/readelf[0x41349b]
/home/evilsocket/binutils-2.25/binutils/readelf[0x490b67]
/home/evilsocket/binutils-2.25/binutils/readelf[0x4c0b44]
/home/evilsocket/binutils-2.25/binutils/readelf[0x403b0d]
/lib/x86_64-linux-gnu/libc.so.6(__libc_start_main+0xf0)[0x7ffff7814a40]
/home/evilsocket/binutils-2.25/binutils/readelf[0x404169]
======= Memory map: ========
00400000-0058b000 r-xp 00000000 08:05 6179323
/home/evilsocket/binutils-2.25/binutils/readelf
0078a000-0078b000 r--p 0018a000 08:05 6179323
/home/evilsocket/binutils-2.25/binutils/readelf
0078b000-0078d000 rw-p 0018b000 08:05 6179323
/home/evilsocket/binutils-2.25/binutils/readelf
0078d000-007b1000 rw-p 00000000 00:00 0
[heap]
7ffff7313000-7ffff7329000 r-xp 00000000 08:05 262358
/lib/x86_64-linux-gnu/libgcc_s.so.1
7ffff7329000-7ffff7528000 ---p 00016000 08:05 262358
/lib/x86_64-linux-gnu/libgcc_s.so.1
7ffff7528000-7ffff7529000 rw-p 00015000 08:05 262358
/lib/x86_64-linux-gnu/libgcc_s.so.1
7ffff7529000-7ffff77f4000 r--p 00000000 08:05 6553617
/usr/lib/locale/locale-archive
7ffff77f4000-7ffff79b4000 r-xp 00000000 08:05 262349
/lib/x86_64-linux-gnu/libc-2.21.so
7ffff79b4000-7ffff7bb4000 ---p 001c0000 08:05 262349
/lib/x86_64-linux-gnu/libc-2.21.so
7ffff7bb4000-7ffff7bb8000 r--p 001c0000 08:05 262349
/lib/x86_64-linux-gnu/libc-2.21.so
7ffff7bb8000-7ffff7bba000 rw-p 001c4000 08:05 262349
/lib/x86_64-linux-gnu/libc-2.21.so
7ffff7bba000-7ffff7bbe000 rw-p 00000000 00:00 0
7ffff7bbe000-7ffff7bd7000 r-xp 00000000 08:05 262234
```

```
/lib/x86_64-linux-gnu/libz.so.1.2.8
7ffff7bd7000-7ffff7dd7000 ---p 00019000 08:05 262234
/lib/x86_64-linux-gnu/libz.so.1.2.8
7ffff7dd7000-7ffff7dd8000 r--p 00019000 08:05 262234
/lib/x86_64-linux-gnu/libz.so.1.2.8
7ffff7dd8000-7ffff7dd9000 rw-p 0001a000 08:05 262234
/lib/x86_64-linux-gnu/libz.so.1.2.8
7ffff7dd9000-7ffff7dfd000 r-xp 00000000 08:05 262335
/lib/x86_64-linux-gnu/ld-2.21.so
7ffff7fa7000-7ffff7fce000 r--p 00000000 08:05 7605558
/usr/share/locale-langpack/it/LC_MESSAGES/binutils.mo
7ffff7fce000-7ffff7fd1000 rw-p 00000000 00:00 0
7ffff7fec000-7ffff7fee000 rw-p 00000000 00:00 0
7ffff7fee000-7ffff7ff5000 r--s 00000000 08:05 6824223
/usr/lib/x86_64-linux-gnu/gconv/gconv-modules.cache
7ffff7ff5000-7ffff7ff8000 rw-p 00000000 00:00 0
7ffff7ff8000-7ffff7ffa000 r--p 00000000 00:00 0
[vvar]
7ffff7ffa000-7ffff7ffc000 r-xp 00000000 00:00 0
[vdso]
7ffff7ffc000-7ffff7ffd000 r--p 00023000 08:05 262335
/lib/x86_64-linux-gnu/ld-2.21.so
7ffff7ffd000-7ffff7ffe000 rw-p 00024000 08:05 262335
/lib/x86_64-linux-gnu/ld-2.21.so
7ffff7ffe000-7ffff7fff000 rw-p 00000000 00:00 0
7ffffffde000-7ffffffff000 rw-p 00000000 00:00 0
[stack]
ffffffffff600000-ffffffffff601000 r-xp 00000000 00:00 0
[vsyscall]
  0x0010:   nome: GCC_3.0
Program received signal SIGABRT, Aborted.
0x00007ffff7829267 in __GI_raise (sig=sig@entry=6) at
../sysdeps/unix/sysv/linux/raise.c:55
55    ../sysdeps/unix/sysv/linux/raise.c: File o directory non esistente.
```

Let AFL run for a while and it will find all sort of edge cases and potentially exploitable vulnerabilities :)

Let's stay in touch!