# Fuzzing techniques & software vulnerabilities

Xavier Claude Mathieu Fourcroy William Robinet

Conostix S.A.

17th October 2016

## Agenda

## Definition

**Definition**

Automated testing technique which provide unexpected data as input for computer program to detect unanticipated behaviour.

## Source of fuzzing

- Fuzzing is inspired by casual users who:
  - Enter dates where money amount is expected
  - Enter digits where names belong...
- This often result in segfaults, stack overflows...
- A fuzzing test crafts such invalid inputs in order to raise exceptions

## Input validation

```
$ change_password
enter new passord (max 7):
```

## Input validation

```
#include <stdio.h>
#include <stdlib.h>

int main(int arg, char *argv[]) {
        char new_pwd[8];
        char *cur_user = getenv("USER");
        printf("Enter new pass for %s (max 7):", \
                                cur_user);
        scanf("%s", new_pwd);

        printf("New password for user %s: %s\n",\
                        cur_user, new_pwd);
}
```

## Input validation

```
$ ./a.out
Enter new password for xavier (min: 5 char, max 7):12345
New password for user xavier: 12345
```

## Input validation

```
$ ./a.out
Enter new password for xavier (min: 5 char, max 7):12345678
New password for user rminal-emulator/1311-10-yavin_TIME1270233: 123456
```

## Fuzzing benefits

- Every programs contain bugs, we just don't know them yet
- Provide results with little effort
- Reveal bugs that were missed in manual audit or static analysis

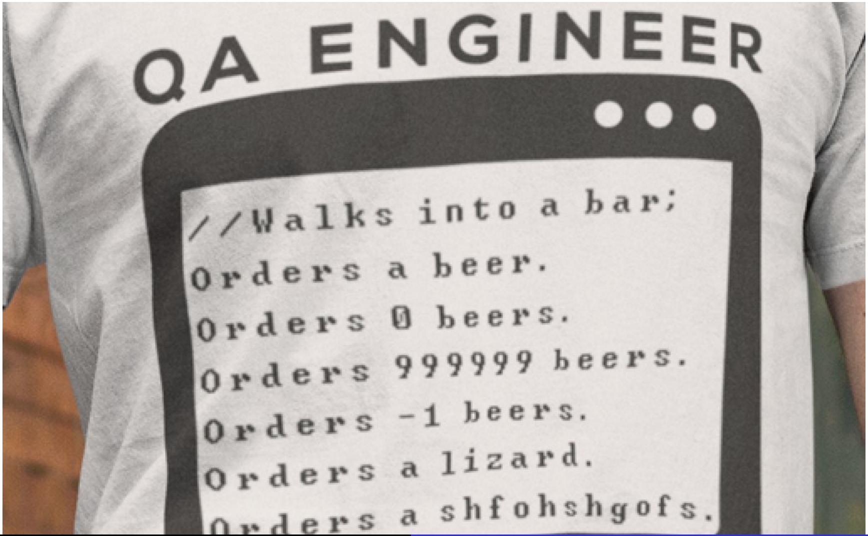
## Fuzzing limitations

- Do not detect all bugs
- Need deeper code investigation to analyse crashing test cases
- Not so easy with programs requiring complex inputs

## Fuzzing techniques

- Manual
- Fully random
- Guided fuzzing

## Manual


QA ENGINEER
//Walks into a bar;
Orders a beer.
Orders 0 beers.
Orders 999999 beers.
Orders -1 beers.
Orders a lizard.
Orders a shfohshgofs.

## Fully random

```
$ bc < /dev/urandom
```

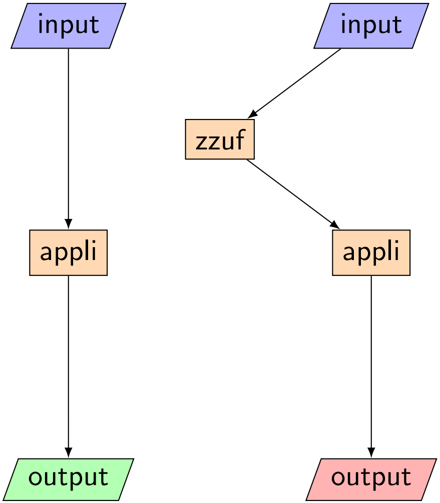
## Guided fuzzing

Analyze program behaviour to adapt fuzzing

## What is zzuf

- https://github.com/samhocevar/zzuf
- Easy-to-use fuzzing software
- Ability to reproduce behaviour
- Can fuzz everything

## What is zzuf



- generates test cases
- records test cases in order to reproduce them
- injects test cases
- intercepts file reading functions
- checks STDOUT and exit values
- detects crashes

## Input generation

- Original file

```
$ cat zzuf_demo_txt
ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
Hello world!!
```

- 3% randomness

```
$ zzuf -r0.03 cat zzuf_demo_txt
ABADEFGHIJKLMVOPURSTUVUXYZ
ab#d%fghihklmnopqrstuvwpyz
01234567:9
Hello world!!
```

- 20% randomness

```
$ zzuf -r0.2 cat zzuf_demo_txt
ARGEEFWHIRYHMNLPQSSTUVWXQz
s(cdufghijid/nnp0n3Le4wxy:
01R74=.'x)
*}dlo gozdf!!
```

## American fuzzy lop

## Description

- Focus on performance

## Description

- Focus on performance
- Bruteforce with instrumentation guided genetic algorithm and edge coverage

## Description

- Focus on performance
- Bruteforce with instrumentation guided genetic algorithm and edge coverage
- Try to minimize result

## Instrumenting with source

- Use a GCC/Clang wrapper

## Instrumenting with source

- Use a GCC/Clang wrapper
- Add a random identifier for each branch

## Instrumenting with source

- Use a GCC/Clang wrapper
- Add a random identifier for each branch
- Compiler options to detect bad behaviour

## Instrumenting with source

- Use a GCC/Clang wrapper
- Add a random identifier for each branch
- Compiler options to detect bad behaviour

-
  ```
  $ CC=/path/to/afl/afl-gcc ./configure
  $ make clean all
  ```

## Instrumenting blackbox

- Use a modified version of Qemu

## Instrumenting blackbox

- Use a modified version of Qemu
- Slower than the source instrumentation

## Instrumenting blackbox

- Use a modified version of Qemu
- Slower than the source instrumentation
- Doesn't require source

## Code coverage

- Record each branch jump with a random id

```
cur_location = <COMPILE_TIME_RANDOM>;
shared_mem[cur_location ^ prev_location]++;
prev_location = cur_location >> 1;
```

## Code coverage

- Record each branch jump with a random id

```
cur_location = <COMPILE_TIME_RANDOM>;
shared_mem[cur_location ^ prev_location]++;
prev_location = cur_location >> 1;
```

- This works well on "standard" program (< 10k branch)

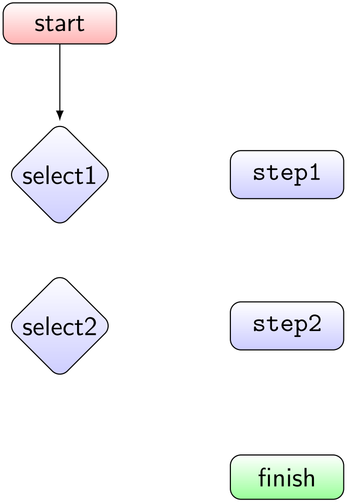
## Code coverage

- Record each branch jump with a random id

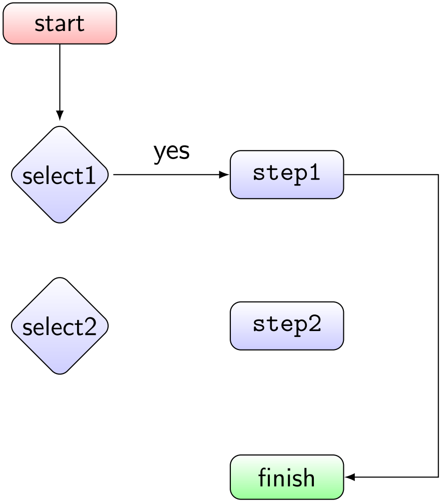
```
cur_location = <COMPILE_TIME_RANDOM>;
shared_mem[cur_location ^ prev_location]++;
prev_location = cur_location >> 1;
```

- This works well on "standard" program ($<$ 10k branch)
- This allows a fast lookup (limit perf impact during fuzzing)

## Path discovery

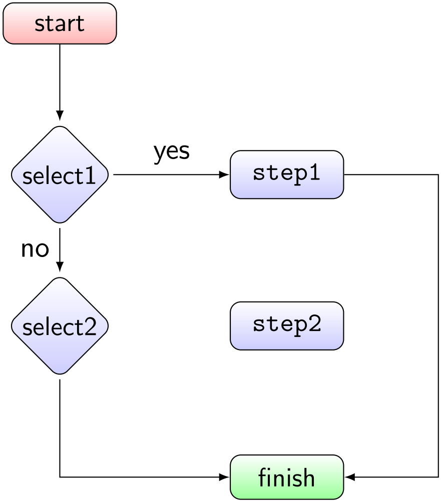


## Path discovery



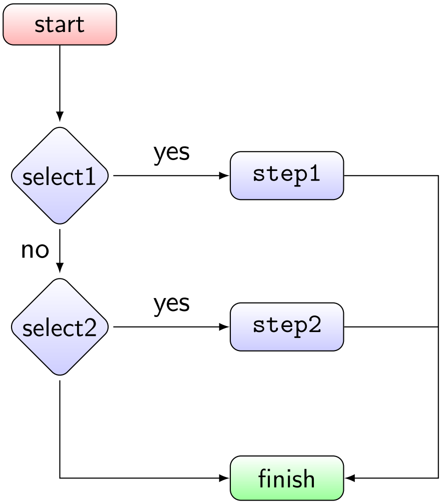
Paths: 1

## Path discovery



Paths: 2

## Path discovery



Paths: 3

## Address sanitizer

- Compiler extension to find invalid memory management

## Address sanitizer

- Compiler extension to find invalid memory management
- Lot of memory consumption (20TB)

## Example

```
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    FILE *fp;
    char buff[16];

    fp = fopen("/tmp/test.txt", "r");
    fscanf(fp, "%s", buff);
    fclose(fp);

    if(buff[0] == 0x66)
        if(buff[1] == 0x6f)
            if(buff[2] == 0x6f) {
                printf("Password accepted\n");
                abort();
            }

    if(buff[0] == 0x00)
        printf("Password empty\n");

    return 0;
}
```

## Compilation

-
  ```
  $ ./afl-gcc -o tests/test ~/projects/centr-conf/src/testafl.c
  afl-cc 2.35b by <lcamtuf@google.com>
  afl-as 2.35b by <lcamtuf@google.com>
  [+] Instrumented 7 locations (64-bit, non-hardened mode, ratio 100
  ```

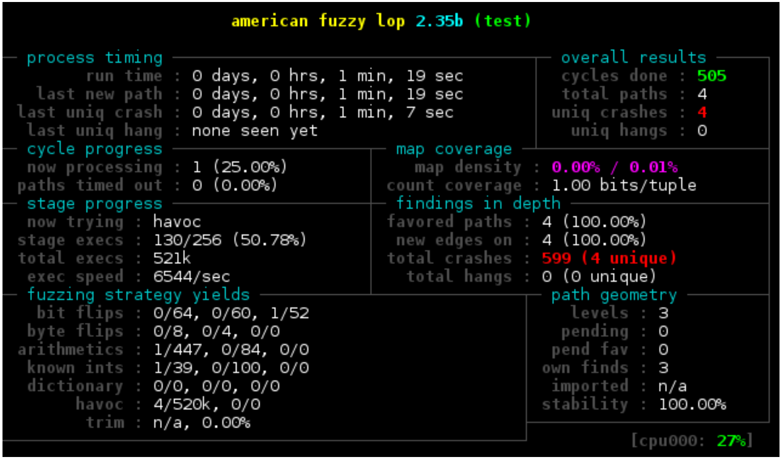
## Compilation

-
  ```
  $ ./afl-gcc -o tests/test ~/projects/centr-conf/src/testafl.c
  afl-cc 2.35b by <lcamtuf@google.com>
  afl-as 2.35b by <lcamtuf@google.com>
  [+] Instrumented 7 locations (64-bit, non-hardened mode, ratio 100%
  ```
- Creating test file:

  ```
  $ echo 'a' > in_test/in
  ```

## Running

```
$ ./afl-fuzz -i in_test/ -o out_test/ -f /tmp/test.txt -- ./tests/test
```



```
                    american fuzzy lop 2.35b (test)

┌─ process timing ─────────────────────────────────────┬─ overall results ─────┐
│        run time : 0 days, 0 hrs, 1 min, 19 sec        │  cycles done : 505    │
│   last new path : 0 days, 0 hrs, 1 min, 19 sec        │  total paths : 4      │
│ last uniq crash : 0 days, 0 hrs, 1 min, 7 sec         │ uniq crashes : 4      │
│  last uniq hang : none seen yet                       │   uniq hangs : 0      │
├─ cycle progress ────────────────────┬─ map coverage ─┴───────────────────────┤
│  now processing : 1 (25.00%)        │    map density : 0.00% / 0.01%         │
│ paths timed out : 0 (0.00%)         │ count coverage : 1.00 bits/tuple       │
├─ stage progress ────────────────────┼─ findings in depth ────────────────────┤
│  now trying : havoc                 │ favored paths : 4 (100.00%)            │
│ stage execs : 130/256 (50.78%)      │  new edges on : 4 (100.00%)            │
│ total execs : 521k                  │ total crashes : 599 (4 unique)         │
│  exec speed : 6544/sec              │   total hangs : 0 (0 unique)           │
├─ fuzzing strategy yields ───────────┴───────────────┬─ path geometry ────────┤
│   bit flips : 0/64, 0/60, 1/52                      │    levels : 3          │
│  byte flips : 0/8, 0/4, 0/0                         │   pending : 0          │
│ arithmetics : 1/447, 0/84, 0/0                      │  pend fav : 0          │
│  known ints : 1/39, 0/100, 0/0                      │ own finds : 3          │
│  dictionary : 0/0, 0/0, 0/0                         │  imported : n/a        │
│       havoc : 4/520k, 0/0                           │ stability : 100.00%    │
│        trim : n/a, 0.00%                            ├────────────────────────┘
└─────────────────────────────────────────────────────┘          [cpu000: 27%]
```

## CVE

- CVE-2015-1315 - Info-ZIP UnZip - Out-of-bounds Write
  `http://www.openwall.com/lists/oss-security/2015/02/17/4`

- CVE-2015-3228 - Ghostscript - Integer overflow
  `http://openwall.com/lists/oss-security/2015/07/23/14`

- CVE-2015-1802: bdfReadProperties: property count needs range check
- CVE-2015-1803: bdfReadCharacters: bailout if a char's bitmap cannot be read
- CVE-2015-1804: bdfReadCharacters: ensure metrics fit into xCharInfo struct
  `https://www.x.org/wiki/Development/Security/Advisory-2015-03-17/`

- CVE-2015-1845, CVE-2015-1846 - unzoo - Buffer overflow & Infinite loop
  `http://seclists.org/oss-sec/2015/q2/4`

- CVE-2014-8130 libtiff: Divide By Zero in the tiffdither tool
  `http://bugzilla.maptools.org/show_bug.cgi?id=2483`
- CVE-2014-8127 libtiff: Out-of-bounds Read in the thumbnail tool
  `http://bugzilla.maptools.org/show_bug.cgi?id=2484`
- CVE-2014-8127 libtiff: Out-of-bounds Read in the tiff2bw tool
  `http://bugzilla.maptools.org/show_bug.cgi?id=2485`
- CVE-2014-8127 libtiff: Out-of-bounds Read in the tiff2rgba tool
  `http://bugzilla.maptools.org/show_bug.cgi?id=2486`
- CVE-2014-8129 libtiff: Out-of-bounds Read & Write in the tiff2pdf tool
  `http://bugzilla.maptools.org/show_bug.cgi?id=2487`

- CVE-2014-8129 libtiff: Out-of-bounds Read & Write in the tiff2pdf tool
  `http://bugzilla.maptools.org/show_bug.cgi?id=2488`
- CVE-2014-8128 libtiff: Out-of-bounds Write in the thumbnail tool
  `http://bugzilla.maptools.org/show_bug.cgi?id=2489`
- CVE-2014-8128 libtiff: Out-of-bounds Write in the tiffdither tool
  `http://bugzilla.maptools.org/show_bug.cgi?id=2490`
- CVE-2014-8128 libtiff: Out-of-bounds Write in the tiffdither tool
  `http://bugzilla.maptools.org/show_bug.cgi?id=2491`
- CVE-2014-8128 libtiff: Out-of-bounds Write in the tiffdither tool
  `http://bugzilla.maptools.org/show_bug.cgi?id=2492`
- CVE-2014-8128 libtiff: Out-of-bounds Write in the thumbnail and tiffcmp tools
  `http://bugzilla.maptools.org/show_bug.cgi?id=2493`
- CVE-2014-8128 libtiff: Out-of-bounds Write in the tiff2pdf tool
  `http://bugzilla.maptools.org/show_bug.cgi?id=2495`
- CVE-2014-8127 libtiff: Out-of-bounds Read in the tiff2ps and tiffdither tools
  `http://bugzilla.maptools.org/show_bug.cgi?id=2496`
- CVE-2014-8127 libtiff: Out-of-bounds Read in the tiffmedian tool
  `http://bugzilla.maptools.org/show_bug.cgi?id=2497`
- CVE-2014-8128 libtiff: Out-of-bounds Write in the thumbnail and tiffcmp tools
  `http://bugzilla.maptools.org/show_bug.cgi?id=2499`
- CVE-2014-8127 libtiff: Out-of-bounds Read in the tiffset tool
  `http://bugzilla.maptools.org/show_bug.cgi?id=2500`
- CVE-2014-8128 libtiff: Out-of-bounds Writes in the tiffdither tool
  `http://bugzilla.maptools.org/show_bug.cgi?id=2501`

## Upstream

- Some developpers welcome any bug report

## Upstream

- Some developpers welcome any bug report
- Others doesn't like when the program is not used as intended

## Upstream

- Some developpers welcome any bug report
- Others doesn't like when the program is not used as intended
- Most doesn't answer at all

## Helping fuzzer

- Allow entry points everywhere in the software

## Helping fuzzer

- Allow entry points everywhere in the software
- Allow input file/stdin for every file

## Conclusion

Thank you for listening!

Useful links:

- AFL: `http://lcamtuf.coredump.cx/afl/`
- The Fuzzing Project: `https://fuzzing-project.org/`