Fuzzers

Team 9

Shashank Chitti
Jose Lemus
Anthony Saccamano
Arlyn Rodriguez
Alex Wong

Outline

- Introduction to fuzzing
- American Fuzzy Lop and ZZUF
- Project Work
- Compare and Contrast

Intro to Fuzzers

- Inject random data into program to detect bugs
- Purpose to see if program can handle unintended input
- Pros
  - Source code not required (but useful!)
  - Lower effort
  - Good at finding implementation issues (SQL, overflows, memory corruption)

Cons

- Not Accurate (tries many different cases)
- May require considerable time and computing resources.
- Bad at finding logic issues.

We've been working with

AFL (American Fuzzy Lop)

ZZUF

American Fuzzy Lop (AFL)

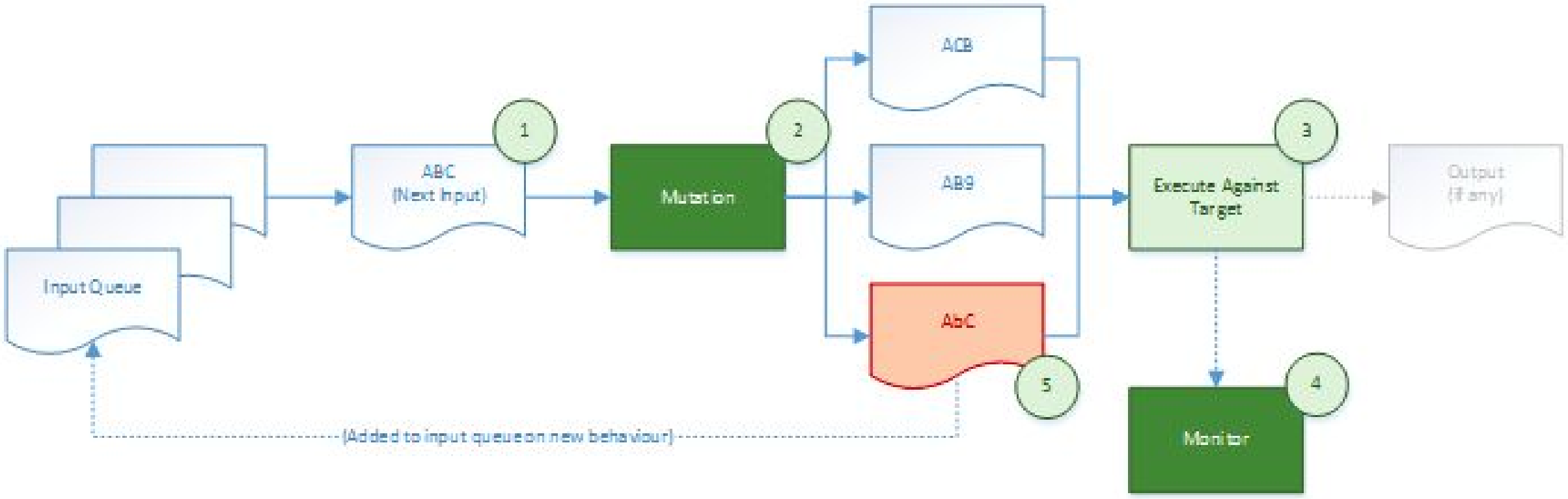
- Brute Force Fuzzer with instrumentation guided algorithm.
- Made by Michael Zalewiski from Google in 2013.
- Different from other fuzzers because it can work with source code and binaries.
- Kinda smart, but still kinda dumb :).

Real time output!

```
                    american fuzzy lop 1.95b (tar)

 process timing                                          overall results
        run time : 0 days, 0 hrs, 1 min, 26 sec          cycles done : 0
   last new path : 0 days, 0 hrs, 0 min, 41 sec          total paths : 21
 last uniq crash : none seen yet                        uniq crashes : 0
  last uniq hang : none seen yet                          uniq hangs : 0
 cycle progress                          map coverage
  now processing : 2 (9.52%)               map density : 1004 (1.53%)
 paths timed out : 0 (0.00%)            count coverage : 1.19 bits/tuple
 stage progress                          findings in depth
  now trying : arith 8/8                favored paths : 12 (57.14%)
 stage execs : 4128/9015 (45.79%)        new edges on : 16 (76.19%)
 total execs : 53.0k                    total crashes : 0 (0 unique)
  exec speed : 634.4/sec                  total hangs : 0 (0 unique)
 fuzzing strategy yields                                 path geometry
   bit flips : 10/2720, 0/2717, 4/2711                     levels : 2
  byte flips : 0/340, 0/337, 0/331                        pending : 19
 arithmetics : 2/10.9k, 0/2449, 0/391                    pend fav : 11
  known ints : 0/1214, 3/5170, 1/8207                   own finds : 20
  dictionary : 0/0, 0/0, 0/858                           imported : n/a
       havoc : 0/10.0k, 0/0                              variable : 3
        trim : 42.57%/162, 0.00%
                                                        [cpu:100%]
```

How AFL Works



https://www.nettitude.co.uk/fuzzing-with-american-fuzzy-lop-afl/

Process Timing

```
                    american fuzzy lop 1.95b (tar)

 process timing                                      overall results
        run time : 0 days, 0 hrs, 1 min, 26 sec      cycles done : 0
   last new path : 0 days, 0 hrs, 0 min, 41 sec      total paths : 21
 last uniq crash : none seen yet                    uniq crashes : 0
  last uniq hang : none seen yet                      uniq hangs : 0

  now processing : 2 (9.52%)            map density : 1004 (1.53%)
 paths timed out : 0 (0.00%)         count coverage : 1.19 bits/tuple
 stage progress                       findings in depth
  now trying : arith 8/8             favored paths : 12 (57.14%)
 stage execs : 4128/9015 (45.79%)     new edges on : 16 (76.19%)
 total execs : 53.0k                total crashes : 0 (0 unique)
  exec speed : 634.4/sec              total hangs : 0 (0 unique)
 fuzzing strategy yields                            path geometry
   bit flips : 10/2720, 0/2717, 4/2711                levels : 2
  byte flips : 0/340, 0/337, 0/331                   pending : 19
 arithmetics : 2/10.9k, 0/2449, 0/391               pend fav : 11
  known ints : 0/1214, 3/5170, 1/8207              own finds : 20
  dictionary : 0/0, 0/0, 0/858                      imported : n/a
       havoc : 0/10.0k, 0/0                         variable : 3
        trim : 42.57%/162, 0.00%
                                                          [cpu:100%]
```

Run time: How long the fuzzer has been running.

Last new path: How long ago the fuzzer found a new path in the program.

Last uniq crash: Last time the fuzzer made the program crash.

Last uniq Hang: Last time the fuzzer had a unique hang.

Overall Results

```
                    american fuzzy lop 1.95b (tar)

┌─ process timing ─────────────────────────────────────┬─ overall results ─────┐
│        run time : 0 days, 0 hrs, 1 min, 26 sec       │  cycles done : 0      │
│   last new path : 0 days, 0 hrs, 0 min, 41 sec       │  total paths : 21     │
│ last uniq crash : none seen yet                      │ uniq crashes : 0      │
│  last uniq hang : none seen yet                      │   uniq hangs : 0      │
├─ cycle progress ────────────────────┬─ map coverage ─┴───────────────────────┤
│  now processing : 2 (9.52%)         │    map density : 1004 (1.53%)          │
│ paths timed out : 0 (0.00%)         │ count coverage : 1.19 bits/tuple       │
├─ stage progress ────────────────────┼─ findings in depth ────────────────────┤
│  now trying : arith 8/8             │ favored paths : 12 (57.14%)            │
│ stage execs : 4128/9015 (45.79%)    │  new edges on : 16 (76.19%)            │
│ total execs : 53.0k                 │ total crashes : 0 (0 unique)           │
│  exec speed : 634.4/sec             │   total hangs : 0 (0 unique)           │
├─ fuzzing strategy yields ───────────┴───────────────┬─ path geometry ────────┤
│   bit flips : 10/2720, 0/2717, 4/2711               │    levels : 2          │
│  byte flips : 0/340, 0/337, 0/331                   │   pending : 19         │
│ arithmetics : 2/10.9k, 0/2449, 0/391                │  pend fav : 11         │
│  known ints : 0/1214, 3/5170, 1/8207                │ own finds : 20         │
│  dictionary : 0/0, 0/0, 0/858                       │  imported : n/a        │
│       havoc : 0/10.0k, 0/0                          │  variable : 3          │
│        trim : 42.57%/162, 0.00%                     ├────────────────────────┘
└─────────────────────────────────────────────────────┘          [cpu:100%]
```

Cyles done: # of times afl went over all interesting test cases and fuzzed them.

Total paths: # of inputs that have uncovered unique paths in the code.

Uniq crashes: When the program encounters a bug, this counter will increment. Time to celebrate!

Uniq Hangs: How many times the program has hanged on a given input.

Stage Progress

```
                    american fuzzy lop 1.95b (tar)

 process timing                                      overall results
        run time : 0 days, 0 hrs, 1 min, 26 sec      cycles done : 0
   last new path : 0 days, 0 hrs, 0 min, 41 sec      total paths : 21
 last uniq crash : none seen yet                    uniq crashes : 0
  last uniq hang : none seen yet                      uniq hangs : 0
 cycle progress                        map coverage
  now processing : 2 (9.52%)             map density : 1004 (1.53%)
                                        ount coverage : 1.19 bits/tuple
 stage progress                         findings in depth
  now trying : arith 8/8                avored paths : 12 (57.14%)
 stage execs : 4128/9015 (45.79%)        new edges on : 16 (76.19%)
 total execs : 53.0k                    otal crashes : 0 (0 unique)
                                          total hangs : 0 (0 unique)
 fuzzing strategy yields                             path geometry
   bit flips : 10/2720, 0/2717, 4/2711                 levels : 2
  byte flips : 0/340, 0/337, 0/331                    pending : 19
 arithmetics : 2/10.9k, 0/2449, 0/391                pend fav : 11
  known ints : 0/1214, 3/5170, 1/8207                own finds : 20
  dictionary : 0/0, 0/0, 0/858                        imported : n/a
       havoc : 0/10.0k, 0/0                           variable : 3
        trim : 42.57%/162, 0.00%
                                                          [cpu:100%]
```

- now trying: What algorithm the fuzzer is trying currently.
- stage execs: How many executions have been done in the current stage
- total execs - How many executions have been done in total.

Fuzzing Strategy Yields

```
                    american fuzzy lop 1.95b (tar)

┌─ process timing ─────────────────────────────────────┬─ overall results ─────┐
│        run time : 0 days, 0 hrs, 1 min, 26 sec       │  cycles done : 0      │
│   last new path : 0 days, 0 hrs, 0 min, 41 sec       │  total paths : 21     │
│ last uniq crash : none seen yet                      │ uniq crashes : 0      │
│  last uniq hang : none seen yet                      │   uniq hangs : 0      │
├─ cycle progress ────────────────────┬─ map coverage ─┴───────────────────────┤
│  now processing : 2 (9.52%)         │    map density : 1004 (1.53%)          │
│ paths timed out : 0 (0.00%)         │ count coverage : 1.19 bits/tuple       │
├─ stage progress ────────────────────┼─ findings in depth ────────────────────┤
│  now trying : arith 8/8             │ favored paths : 12 (57.14%)            │
│ stage execs : 4128/9015 (45.79%)    │  new edges on : 16 (76.19%)            │
│                                     │ l crashes : 0 (0 unique)               │
│                                     │ tal hangs : 0 (0 unique)               │
├─ fuzzing strategy yields ───────────┴───────────────┬─ path geometry ────────┤
│   bit flips : 10/2720, 0/2717, 4/2711               │    levels : 2          │
│  byte flips : 0/340, 0/337, 0/331                   │   pending : 19         │
│ arithmetics : 2/10.9k, 0/2449, 0/391                │  pend fav : 11         │
│  known ints : 0/1214, 3/5170, 1/8207                │ own finds : 20         │
│  dictionary : 0/0, 0/0, 0/858                       │  imported : n/a        │
│       havoc : 0/10.0k, 0/0                          │  variable : 3          │
│        trim : 42.57%/162, 0.00%                     ├────────────────────────┘
└─────────────────────────────────────────────────────┘             [cpu:100%]
```

Input Generation

- Uses multiple mutation algorithms on each fuzzing cycle
  - Bit-flips, substitutions, randomized data, etc.
- Identifies favored paths and marks for further fuzzing
  - Favored path: mutation that causes new change in control flow (i.e. conditional branches)
  - afl-fuzz enqueues mutations that result in favored paths as inputs for future fuzzing
- Intelligently mutates input based on previous cycles
  - Cycle 0: Start with given input, run algorithms
    - If input looks promising, put input into queue
  - Cycle 1+: Analyze results of previous cycle(s) and choose new input
    - Input from queue or from most recent cycles

AFL Mutation Algorithms

- bitflp L/S
  - Every *S* bits, flip *L* bits (1 <-> 0)
  - Deterministic (non-random) = repeatable!
  - Ex: 16/8 = Choose every 8th bit in the file, and flip 16 of them.
  - Fixed L/S pairs; iterate through all pairs each cycle
    - 1/1, 2/1, 4/1, 8/8, 16/8, 32/8
- arith L/8
  - Identifies 8-, 16-, and 32-bit numbers
  - Adds or subtracts small, deterministic values to these numbers
  - Steps of 8 bits (i.e. check for values every 8 bits)

Source: http://lcamtuf.coredump.cx/afl/status_screen.txt

AFL Mutation Algorithms

- interest L/8
  - Similar to arith L/8, but values overwritten instead of added/subtracted
  - Uses “interesting” 8-, 16-, and 32-bit values
    - Corner cases, potential overflows, etc.
    - EX: 8-bit interesting values: 0, 255
- extras
  - Inject specific terms into input
  - Sourced from user-defined dictionary and/or automatically-generated dictionary
  - Tests with overwriting and inserting

Source: http://lcamtuf.coredump.cx/afl/status_screen.txt

AFL Mutation Algorithms

- havoc
  - Fixed cycle limit
  - Random mutations
    - Bit flips
    - Overwrites
    - Data deletion and/or duplication
    - Random dictionary operations (if dictionary is user-defined)
- splice
  - Last-resort algorithm
  - Invoked in first cycle after no new paths found
  - Similar to havoc
    - Randomly-selected algorithms
    - Input is a “splice” of two randomly-selected inputs from the queue

Source: http://lcamtuf.coredump.cx/afl/status_screen.txt

Fuzzing coreutils + binutils

- GNU Core Utilities (coreutils), version 8.24
  - Basic CLI tools
  - Installed on nearly every Unix-like OS
  - Ex: cat, ls, rm, chmod, etc.

Fuzzing: head and md5sum

- GNU Binary Utilities (binutils), version 2.25
  - Programming tools
  - Operate on binary files, object files, assembly, etc.
  - Ex: objdump, readelf

Fuzzing: objdump, size, strip, readelf, strings, nm

Results
Readelf (binutils)

```
                    american fuzzy lop 1.85b (readelf)

┌─ process timing ─────────────────────────────────────┬─ overall results ─────┐
│        run time : 5 days, 21 hrs, 3 min, 18 sec      │  cycles done : 0      │
│   last new path : 0 days, 2 hrs, 34 min, 1 sec       │  total paths : 2251   │
│ last uniq crash : none seen yet                      │ uniq crashes : 0      │
│  last uniq hang : 0 days, 1 hrs, 40 min, 30 sec      │   uniq hangs : 82     │
├─ cycle progress ────────────────────┬─ map coverage ─┴───────────────────────┤
│  now processing : 296 (13.15%)      │    map density : 6122 (9.34%)          │
│ paths timed out : 0 (0.00%)         │ count coverage : 2.57 bits/tuple       │
├─ stage progress ────────────────────┼─ findings in depth ────────────────────┤
│  now trying : arith 32/8            │ favored paths : 676 (30.03%)           │
│ stage execs : 38.9k/12.9M (0.30%)   │  new edges on : 946 (42.03%)           │
│ total execs : 215M                  │ total crashes : 0 (0 unique)           │
│  exec speed : 384.9/sec             │   total hangs : 7325 (82 unique)       │
├─ fuzzing strategy yields ───────────┴───────────────┬─ path geometry ────────┤
│   bit flips : 759/43.2M, 109/43.2M, 109/43.2M       │    levels : 3          │
│  byte flips : 16/5.40M, 4/361k, 2/393k              │   pending : 2193       │
│ arithmetics : 420/19.3M, 21/18.4M, 1/15.6M          │  pend fav : 642        │
│  known ints : 27/1.04M, 70/4.74M, 55/9.63M          │ own finds : 2250       │
│  dictionary : 0/0, 0/0, 56/9.12M                    │  imported : n/a        │
│       havoc : 510/1.78M, 0/0                        │  variable : 0          │
│        trim : 1.69%/83.5k, 93.59%                   ├────────────────────────┘
└─────────────────────────────────────────────────────┘             [cpu:214%]
```

- As of 11/8, still no bugs in ReadElf.

Results
strings (binutils)

```
                    american fuzzy lop 1.85b (strings)

┌─ process timing ─────────────────────────────────────┬─ overall results ─────┐
│        run time : 5 days, 20 hrs, 55 min, 33 sec     │  cycles done : 6197   │
│   last new path : 5 days, 20 hrs, 33 min, 23 sec     │  total paths : 92     │
│ last uniq crash : none seen yet                      │ uniq crashes : 0      │
│  last uniq hang : 5 days, 11 hrs, 10 min, 26 sec     │   uniq hangs : 7      │
├─ cycle progress ────────────────────┬─ map coverage ─┴───────────────────────┤
│  now processing : 56* (60.87%)      │    map density : 87 (0.13%)            │
│ paths timed out : 0 (0.00%)         │ count coverage : 4.76 bits/tuple       │
├─ stage progress ────────────────────┼─ findings in depth ────────────────────┤
│  now trying : splice 4              │ favored paths : 5 (5.43%)              │
│ stage execs : 210/500 (42.00%)      │  new edges on : 9 (9.78%)              │
│ total execs : 757M                  │ total crashes : 0 (0 unique)           │
│  exec speed : 1084/sec              │   total hangs : 268 (7 unique)         │
├─ fuzzing strategy yields ───────────┴───────────────┬─ path geometry ────────┤
│   bit flips : 8/720k, 0/720k, 0/720k                │    levels : 4          │
│  byte flips : 0/90.1k, 2/55.1k, 2/56.1k             │   pending : 0          │
│ arithmetics : 0/3.05M, 0/1.34M, 0/109k              │  pend fav : 0          │
│  known ints : 0/288k, 1/1.48M, 2/2.43M              │ own finds : 91         │
│  dictionary : 0/0, 0/0, 0/44.8k                     │  imported : n/a        │
│       havoc : 56/278M, 19/467M                      │  variable : 0          │
│        trim : 3.36%/17.4k, 39.27%                   ├────────────────────────┘
└─────────────────────────────────────────────────────┘             [cpu:200%]
```

- As of 11/8, still no bugs in strings.

Results
nm (binutils, QEMU mode)

```
                    american fuzzy lop 1.85b (nm)

┌─ process timing ─────────────────────────────────────┬─ overall results ─────┐
│        run time : 2 days, 1 hrs, 20 min, 3 sec       │  cycles done : 8      │
│   last new path : 0 days, 5 hrs, 13 min, 8 sec       │  total paths : 77     │
│ last uniq crash : 2 days, 1 hrs, 12 min, 16 sec      │ uniq crashes : 2      │
│  last uniq hang : 0 days, 7 hrs, 25 min, 18 sec      │   uniq hangs : 15     │
├─ cycle progress ────────────────────┬─ map coverage ─┴───────────────────────┤
│  now processing : 24* (31.17%)      │    map density : 276 (0.42%)           │
│ paths timed out : 0 (0.00%)         │ count coverage : 1.84 bits/tuple       │
├─ stage progress ────────────────────┼─ findings in depth ────────────────────┤
│  now trying : arith 32/8            │ favored paths : 19 (24.68%)            │
│ stage execs : 17.3k/263k (6.55%)    │  new edges on : 23 (29.87%)            │
│ total execs : 34.4M                 │ total crashes : 177k (2 unique)        │
│  exec speed : 168.9/sec             │   total hangs : 574 (15 unique)        │
├─ fuzzing strategy yields ───────────┴───────────────┬─ path geometry ────────┤
│   bit flips : 34/4.10M, 2/4.10M, 2/4.10M            │    levels : 4          │
│  byte flips : 0/512k, 0/84.2k, 0/96.9k              │   pending : 11         │
│ arithmetics : 21/4.36M, 0/5.18M, 0/4.73M            │  pend fav : 0          │
│  known ints : 4/190k, 1/839k, 1/1.97M               │ own finds : 75         │
│  dictionary : 0/0, 0/0, 4/2.88M                     │  imported : n/a        │
│       havoc : 1/541k, 0/601k                        │  variable : 0          │
│        trim : 1.84%/126k, 84.80%                    ├────────────────────────┘
└─────────────────────────────────────────────────────┘             [cpu:150%]

[nm_simple 0:../afl-1.85b/afl-fuzz*                    "instance-2" 01:08 08-Dec-15
```

- Input file was vuln2.c and a small helloworld.c file.
- Memory exhaust error.
- As of 11/8

Results
objdump (binutils, QEMU mode)

```
                    american fuzzy lop 1.85b (objdump)

┌─ process timing ─────────────────────────────────────┬─ overall results ─────┐
│        run time : 1 days, 1 hrs, 22 min, 37 sec      │  cycles done : 0      │
│   last new path : 0 days, 0 hrs, 41 min, 58 sec      │  total paths : 718    │
│ last uniq crash : 0 days, 7 hrs, 2 min, 50 sec       │ uniq crashes : 9      │
│  last uniq hang : 0 days, 6 hrs, 7 min, 25 sec       │   uniq hangs : 23     │
├─ cycle progress ────────────────────┬─ map coverage ─┴───────────────────────┤
│  now processing : 0 (0.00%)         │    map density : 5363 (8.18%)          │
│ paths timed out : 0 (0.00%)         │ count coverage : 1.53 bits/tuple       │
├─ stage progress ────────────────────┼─ findings in depth ────────────────────┤
│  now trying : auto extras (over)    │ favored paths : 1 (0.14%)              │
│ stage execs : 287k/390k (73.56%)    │  new edges on : 485 (67.55%)           │
│ total execs : 1.96M                 │ total crashes : 4099 (9 unique)        │
│  exec speed : 31.78/sec (slow!)     │   total hangs : 469 (23 unique)        │
├─ fuzzing strategy yields ───────────┴───────────────┬─ path geometry ────────┤
│   bit flips : 488/62.5k, 26/62.5k, 26/62.5k         │    levels : 2          │
│  byte flips : 6/7808, 5/7807, 9/7805                │   pending : 718        │
│ arithmetics : 75/436k, 19/379k, 4/294k              │  pend fav : 1          │
│  known ints : 5/25.3k, 14/111k, 11/204k             │ own finds : 717        │
│  dictionary : 0/0, 0/0, 0/0                         │  imported : n/a        │
│       havoc : 0/0, 0/0                              │  variable : 42         │
│        trim : 0.00%/1931, 0.00%                     ├────────────────────────┘
└─────────────────────────────────────────────────────┘             [cpu:100%]
```

- Input file was vuln2.c and a small helloworld.c file.
- Memory exhaust error.
- objdump -x

Fuzzing: coreutils/head

```
                    american fuzzy lop 1.85b (head)

 process timing                                      overall results
        run time : 4 days, 12 hrs, 0 min, 58 sec     cycles done : 858
   last new path : 0 days, 6 hrs, 54 min, 36 sec     total paths : 137
 last uniq crash : none seen yet                    uniq crashes : 0
  last uniq hang : 2 days, 23 hrs, 40 min, 26 sec     uniq hangs : 10
 cycle progress                        map coverage
  now processing : 99 (72.26%)            map density : 191 (0.29%)
 paths timed out : 0 (0.00%)           count coverage : 2.89 bits/tuple
 stage progress                         findings in depth
  now trying : splice 19               favored paths : 24 (17.52%)
 stage execs : 76/1000 (7.60%)          new edges on : 37 (27.01%)
 total execs : 497M                    total crashes : 0 (0 unique)
  exec speed : 969.3/sec                 total hangs : 38 (10 unique)
 fuzzing strategy yields                              path geometry
   bit flips : 12/4.97M, 0/4.97M, 0/4.97M               levels : 6
  byte flips : 0/621k, 0/9882, 0/10.9k                 pending : 0
 arithmetics : 3/522k, 0/107k, 0/9374                 pend fav : 0
  known ints : 0/49.9k, 0/269k, 0/477k                own finds : 136
  dictionary : 0/0, 0/0, 0/0                           imported : n/a
       havoc : 69/163M, 43/317M                        variable : 0
        trim : 10.19%/68.3k, 98.43%
^C                                                          [cpu:307%]
```

- head: outputs first *n* lines of a text file (default 10)
- Input: 11-line .txt file with random ASCII characters
- Results: No crashes, 10 unique hangs

Fuzzing: coreutils/md5sum

```
                    american fuzzy lop 1.85b (md5sum)

┌─ process timing ─────────────────────────────────────┬─ overall results ─────┐
│        run time : 4 days, 11 hrs, 9 min, 4 sec       │  cycles done : 3059   │
│   last new path : 4 days, 11 hrs, 9 min, 1 sec       │  total paths : 14     │
│ last uniq crash : none seen yet                      │ uniq crashes : 0      │
│  last uniq hang : 2 days, 23 hrs, 6 min, 50 sec      │   uniq hangs : 13     │
├─ cycle progress ────────────────────┬─ map coverage ─┴───────────────────────┤
│  now processing : 4 (28.57%)        │    map density : 132 (0.20%)           │
│ paths timed out : 0 (0.00%)         │ count coverage : 1.02 bits/tuple       │
├─ stage progress ────────────────────┼─ findings in depth ────────────────────┤
│  now trying : havoc                 │ favored paths : 13 (92.86%)            │
│ stage execs : 3224/5000 (64.48%)    │  new edges on : 14 (100.00%)           │
│ total execs : 540M                  │ total crashes : 0 (0 unique)           │
│  exec speed : 1316/sec              │   total hangs : 43 (13 unique)         │
├─ fuzzing strategy yields ───────────┴───────────────┬─ path geometry ────────┤
│   bit flips : 0/19.4k, 0/19.4k, 0/19.3k             │    levels : 2          │
│  byte flips : 0/2423, 0/236, 0/231                  │   pending : 0          │
│ arithmetics : 0/13.4k, 0/2613, 0/707                │  pend fav : 0          │
│  known ints : 0/1303, 0/6147, 0/9907                │ own finds : 13         │
│  dictionary : 0/0, 0/0, 0/0                         │  imported : n/a        │
│       havoc : 13/199M, 0/340M                       │  variable : 0          │
│        trim : 0.62%/1044, 88.35%                    ├────────────────────────┘
└─────────────────────────────────────────────────────┘          [cpu:251%]
^C
```

- md5sum: returns MD5 hash of an input file
- Input: 3-line .txt file with random ASCII characters
- Results: No crashes, 13 unique hangs

Fuzzing: binutils/objdump

```
                    american fuzzy lop 1.85b (objdump)

┌─ process timing ─────────────────────────────────────┬─ overall results ─────┐
│        run time : 3 days, 17 hrs, 11 min, 1 sec      │  cycles done : 62.1k  │
│   last new path : none yet (odd, check syntax!)      │  total paths : 1      │
│ last uniq crash : none seen yet                      │ uniq crashes : 0      │
│  last uniq hang : 3 days, 15 hrs, 16 min, 41 sec     │   uniq hangs : 3      │
├─ cycle progress ────────────────────┬─ map coverage ─┴───────────────────────┤
│  now processing : 0 (0.00%)         │    map density : 39 (0.06%)            │
│ paths timed out : 0 (0.00%)         │ count coverage : 1.00 bits/tuple       │
├─ stage progress ────────────────────┼─ findings in depth ────────────────────┤
│  now trying : havoc                 │ favored paths : 1 (100.00%)            │
│ stage execs : 1440/5000 (28.80%)    │  new edges on : 1 (100.00%)            │
│ total execs : 310M                  │ total crashes : 0 (0 unique)           │
│  exec speed : 1503/sec              │   total hangs : 42 (3 unique)          │
├─ fuzzing strategy yields ───────────┴───────────────┬─ path geometry ────────┤
│   bit flips : 0/32, 0/31, 0/29                      │    levels : 1          │
│  byte flips : 0/4, 0/3, 0/1                         │   pending : 0          │
│ arithmetics : 0/224, 0/0, 0/0                       │  pend fav : 0          │
│  known ints : 0/26, 0/82, 0/44                      │ own finds : 0          │
│  dictionary : 0/0, 0/0, 0/0                         │  imported : n/a        │
│       havoc : 0/310M, 0/0                           │  variable : 0          │
│        trim : 99.93%/18, 0.00%                      ├────────────────────────┘
└─────────────────────────────────────────────────────┘          [cpu:203%]
^C
```

- objdump: displays info about binary files
- Input: prog5 (first stack overflow problem from Challenge 5)
  - objdump called with -D flag
- Result: no crashes, 3 unique hangs

Fuzzing: binutils/size

```
                    american fuzzy lop 1.85b (size)

 process timing                                      overall results
        run time : 3 days, 17 hrs, 6 min, 31 sec      cycles done : 10.4k
   last new path : none yet (odd, check syntax!)      total paths : 2
 last uniq crash : none seen yet                     uniq crashes : 0
  last uniq hang : 3 days, 12 hrs, 2 min, 47 sec       uniq hangs : 4
 cycle progress                        map coverage
  now processing : 0 (0.00%)              map density : 34 (0.05%)
 paths timed out : 0 (0.00%)           count coverage : 1.00 bits/tuple
 stage progress                         findings in depth
  now trying : splice 7                favored paths : 1 (50.00%)
 stage execs : 43/500 (8.60%)           new edges on : 1 (50.00%)
 total execs : 311M                    total crashes : 0 (0 unique)
  exec speed : 2202/sec                  total hangs : 34 (4 unique)
 fuzzing strategy yields                              path geometry
   bit flips : 0/64, 0/62, 0/58                         levels : 1
  byte flips : 0/8, 0/6, 0/2                           pending : 0
 arithmetics : 0/448, 0/0, 0/0                        pend fav : 0
  known ints : 0/49, 0/166, 0/88                      own finds : 0
  dictionary : 0/0, 0/0, 0/0                           imported : n/a
       havoc : 0/103M, 0/207M                          variable : 0
        trim : 99.86%/31, 0.00%
^C                                                    [cpu:163%]
```

- size: outputs size of parts of binary files
- Input: prog5
- Results: No crashes, 4 unique hangs

Fuzzing: binutils/strip

```
                    american fuzzy lop 1.85b (strip-new)

 process timing                                      overall results
        run time : 3 days, 17 hrs, 4 min, 50 sec      cycles done : 61.6k
   last new path : none yet (odd, check syntax!)      total paths : 1
 last uniq crash : none seen yet                     uniq crashes : 0
  last uniq hang : 3 days, 13 hrs, 29 min, 21 sec      uniq hangs : 4
 cycle progress                        map coverage
  now processing : 0 (0.00%)              map density : 85 (0.13%)
 paths timed out : 0 (0.00%)           count coverage : 1.00 bits/tuple
 stage progress                        findings in depth
  now trying : havoc                  favored paths : 1 (100.00%)
 stage execs : 4972/5000 (99.44%)      new edges on : 1 (100.00%)
 total execs : 307M                   total crashes : 0 (0 unique)
  exec speed : 2238/sec                 total hangs : 38 (4 unique)
 fuzzing strategy yields                             path geometry
   bit flips : 0/32, 0/31, 0/29                        levels : 1
  byte flips : 0/4, 0/3, 0/1                          pending : 0
 arithmetics : 0/224, 0/0, 0/0                       pend fav : 0
  known ints : 0/26, 0/82, 0/44                      own finds : 0
  dictionary : 0/0, 0/0, 0/0                          imported : n/a
       havoc : 0/307M, 0/0                            variable : 0
        trim : 99.93%/18, 0.00%
^C                                                          [cpu:104%]
```

- strip: strips extra information from binaries (e.x. debugging info)
- Input: prog5
- Results: No crashes, 4 unique hangs

Vuln2.c Demo

Coverage Results for Vuln2.c

ZZUF

What is it?

-It’s an application input fuzzer, made by caca labs.

-Evolved from the streamf***er tool.

-Intercepts file and network operations and changes random bits in a program’s input.

-Used for QA (stress testing), security (seg faults, etc), and code coverage.

- Used primarily for media players, image viewers, and web browsers. Can be used for other programs (system utilities, vuln, etc).

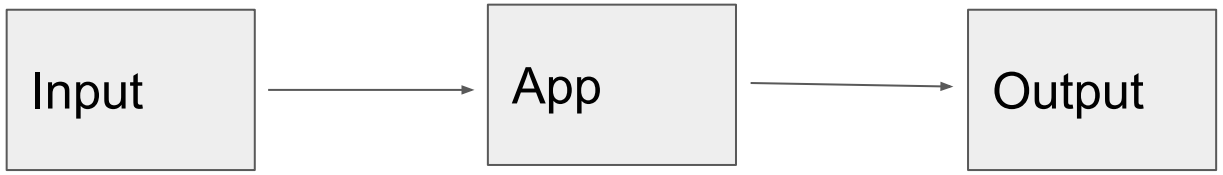
ZZUF Basics

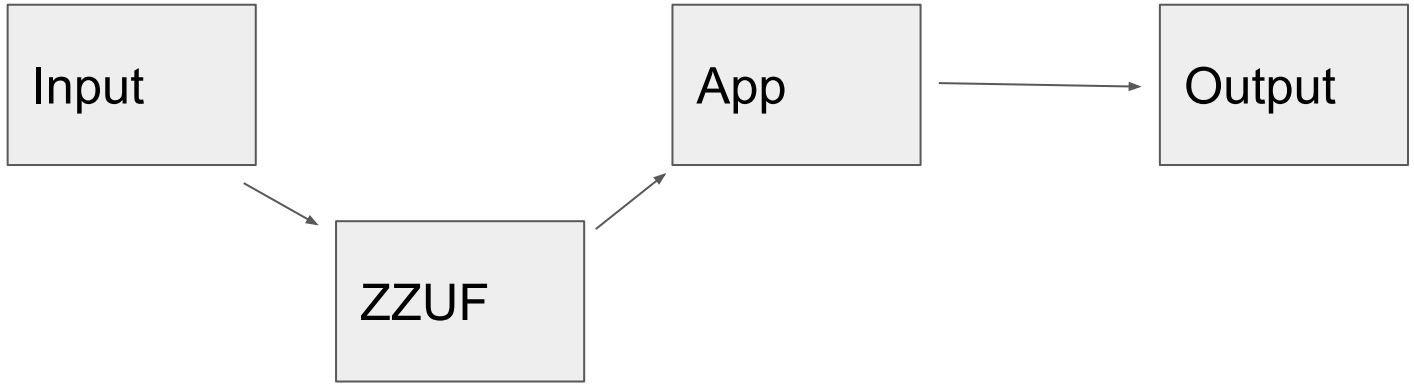
How to use zzuf:

zzuf <flags> <program> <input>

flags that we used:

- -s; seeds=start:stop
- -r; ratio=min:max
- -m; md5 output
- -j; jobs=jobs
- -b; bytes=#ofbytes
- -q; quiet





ZZUF Demo

We'll be demoing 3 things for now:

-ZZUF cat; to show basic functionality

-ZZUF vuln; to show seg fault in action

-ZZUF objdump; to crash objdump (binutils 2.25.1)

ZZUF Demo

Our Project

In addition to just fuzzing random stuff, we decided to compare the two.

Fuzzed some of binutils using both.

Tested

- -Coverage
- -Efficacy

Intro - gcov

- gcov is a test coverage program
- measures which lines of code are executed
  and how many times they are executed
- these results are placed in the .gcov output file generated by gcov
- only works with gcc and the following flags must be added to your gcc command to create the files gcov needs to operate:
  - -fprofile-arcs
  - -ftest-coverage

Example: gcc -fprofile-arcs -ftest-coverage mycode.c -o mycode

Invoking gcov

Must be invoked from the same directory in which gcc was invoked (since all of the metadata needed is in this directory)

```
gcc -fprofile-arcs -ftest-coverage mycode.c -o mycode.o#compile - make sure the .c name is the same as the .o name

        #OR (if you are building a package)

./configure --disable-nls CFLAGS="-g -fprofile-arcs -ftest-coverage" && make  #to build a package

./mycode  #run your code

gcov -f -b mycode.c         #run program from where compiled gcc

lcov --capture --directory . --output-file <lcov-file>.info     #from wherever program was compiled

genhtml <lcov-file>.info
```

Reading gcov output and lcov

The format is

execution count : line number : source line text

'**-**' means that line did not contain any code

'**#####**' means the lines were never executed

Example of part of a .gcov file:

(Note: This .gcov file was created by us after we ran gcov on a modified version of the vuln2 program from Challenge 5)

```
-:    0:Source:vuln2.c
-:    0:Graph:./vuln2.gcno
-:    0:Data:./vuln2.gcda
-:    0:Runs:1
-:    0:Programs:1
-:    1:#include <stdio.h>
-:    2:#include <stdlib.h>
-:    3:#include <string.h>
-:    4:#include <unistd.h>
-:    5:#include <sys/types.h>
-:    6:#include <sys/syscall.h>
-:    7:
-:    8:/*
-:    9: *   print out string with a '--ECHO: ' prefix
-:   10: */
function echo called 1 returned 100% blocks executed 60%
1:   11:void echo(char *s, unsigned int l)
-:   12:{
1:   13:            unsigned char len = (unsigned char) l;
1:   14:            char buf[512] = "--ECHO: ";
-:   15:
1:   16:            strcat(buf, s);
-:   17:
1:   18:            if (len >= 128) {
branch  0 taken 0% (fallthrough)
branch  1 taken 100%
#####:   19:            fprintf(stderr, "argument too long!
\n");
```

gcov results for vuln2 (challenge 5)

LCOV - code coverage report

			Hit	Total	Coverage
Current view:	top level - deliverables				
Test:	lcov-coverage.info	Lines:	19	24	79.2 %
Date:	2015-12-07 03:52:51	Functions:	2	2	100.0 %

Filename	Line Coverage			Functions	
vuln2.c		79.2 %	19 / 24	100.0 %	2 / 2

Generated by: LCOV version 1.11

```
Line data    Source code
       1          : #include <stdio.h>
       2          : #include <stdlib.h>
       3          : #include <string.h>
       4          : #include <unistd.h>
       5          : #include <sys/types.h>
       6          : #include <sys/syscall.h>
       7          :
       8          : /*
       9          :  *   print out string with a '--ECHO: ' prefix
      10          :  */
      11        1 : void echo(char *s, unsigned int l)
      12          : {
      13        1 :                 unsigned char len = (unsigned char) l;
      14        1 :                 char buf[512] = "--ECHO: ";
      15          :
      16        1 :                 strcat(buf, s);
      17          :
      18        1 :                 if (len >= 128) {
      19        0 :                                 fprintf(stderr, "argument too long!\n");
      20        0 :                                 exit(1);
      21          :                 }
      22          :                 else
      23        1 :                                 fprintf(stdout, "%s\n", buf);
      24        1 : }
      25          :
      26          : /*
      27          :  *   simple echo service that prints out its first argument
      28          :  */
```

```
29          1 : int main(int argc, char **argv)
30            : {
31            :                 /* check arguments */
32          1 :                 if (argc != 2) {
33          0 :                                 fprintf(stderr, "please provide one argument to
34          0 :                                 return 1;
35            :                 }
36            :
37            :                 /* call the echo service */
38            :                 FILE *fp;
39            : long lSize;
40            : char *buffer;
41            :
42          1 : fp = fopen ( argv[1] , "rb" );
43          1 : if( !fp ) perror(argv[1]),exit(1);
44            :
45          1 : fseek( fp , 0L , SEEK_END);
46          1 : lSize = ftell( fp );
47          1 : rewind( fp );
48            :
49            : /* allocate memory for entire content */
50          1 : buffer = calloc( 1, lSize+1 );
51          1 : if( !buffer ) fclose(fp),fputs("memory alloc fails",stderr),exit(1);
52            :
53            : /* copy the file into the buffer */
54          1 : if( 1!=fread( buffer , lSize, 1 , fp) )
55          0 :   fclose(fp),free(buffer),fputs("entire read fails",stderr),exit(1);
56          1 : echo(buffer, strlen(buffer));
57            :
58          1 :                 return 0;
59            : }
60            :
```

Results

ZZUF

-nm; memory exhausted. Found a bug in <2 seconds

-objdump; memory exhausted. Found a bug in <5 seconds

AFL

nm; memory exhausted; Found a bug in ~2 hours.

objdump; memory exhausted; Found a bug in ~2 hours.

Differences between AFL and ZZUF

- AFL uses a drop-in replacement for gcc/g++/clang.
- Source code instrumentation feeds AFL coverage information.
- This means that AFL can work on source code and binaries.
- AFL works much better when source code is available due to the instrumentation that it implements
- When no source code, afl uses QEMU emulation
  - Runs 2-5x slower than when source code is available.

Differences between AFL and ZZUF (cont)

- ZZUF cannot inherently be optimized with source code (no special source code instrumentation).
- Almost totally random so compared to AFL it is “dumb”.
- Faster at finding memory faults (so far!).

Present… Future...

-Make a tool to find faults on GNU binutils- So far, we successfully fuzzed two on our own (nm, objdump)

-We will keep it running until our google cloud demo runs out

$200 or 2 months whichever comes first.

Check back for updates on Github!!!

https://github.com/AlexDWong/EC521-fuzzing

Our Tool

Optimized

Relevant input files for different programs in binutils 2.25.1.

- -strings: file with strings
- -nm: object file

Appropriate Flags

- -ez readelf saves first 5 bits for “magic bytes”

Questions?