Zero Bugs Found? Hold My Beer AFL! How to Improve Coverage-guided Fuzzing and Find New Zero-days in Tough Targets

Maksim Shudrak
Security Researcher
Salesforce

DEF CON 27
About me

● Offensive Security Researcher at Salesforce Red Team

● Projects:
  ○ drAFL: AFL + DynamoRIO = fuzzing binaries with no source code on Linux (spare time) https://github.com/mxmssh/drAFL
  ○ Contributions: drltrace, winAFL, DynamoRIO, DrMemory, Ponce
  ○ PhD on vulnerability research in machine code

● Speaker:
Outline

I. Introduction

II. What is coverage-guided fuzzing?

III. Downsides of AFL and similar fuzzers

IV. Introducing Manul

V. DEMO

VI. Case Studies + Vulnerabilities

VII. Conclusion & Future Work
What is Fuzzing?

```c
/* read file */
n = read(buf, BUFSIZE);
if (n < 0)
    error("ERROR in read");

if (buf[0] == 'P') {
    if (buf[1] == 'W') {
        if (buf[2] == 'N') {
            if (buf[3] == 'I') {
                if (buf[4] == 'T') {
                    printf("Found it!\n");
                    ((void(*)())0x0)();
                }
            }
        }
    }
}
```
What is Fuzzing?

```c
/* read file */
int n = read(buf, BUFSIZE);
if (n < 0)
    error("ERROR in read");
if (buf[0] == 'P') {
    if (buf[1] == 'W') {
        if (buf[2] == 'N') {
            if (buf[3] == 'I') {
                if (buf[4] == 'T') {
                    printf("Found it!\n");
                    ((void(*)())0x0)();
                }
            }
        }
    }
}
```
What is Fuzzing?

/* read file */

n = read(buf, BUFSIZE);
if (n < 0)
    error("ERROR in read");

if (buf[0] == 'D') {
    if (buf[1] == 'W') {
        if (buf[2] == 'N') {
            if (buf[3] == 'I') {
                if (buf[4] == 'T') {
                    printf("Found it!\n");
                    ((void(*)(()))0x0)();
                }
            }
        }
    }
}
What is Fuzzing?

/* read file */
n = read(buf, BUFSIZE);
if (n < 0)
    error("ERROR in read");
if (buf[0] == 'D') {
    if (buf[1] == 'W') {
        if (buf[2] == 'N') {
            if (buf[3] == 'I') {
                if (buf[4] == 'T') {
                    printf("Found it!\n");
                    ((void(*)(()))0x0)();
                }
            }
        }
    }
}
What is Fuzzing?

```c
/* read file */
int n = read(buf, BUFSIZE);
if (n < 0)
    error("ERROR in read");

if (buf[0] == 'P') {
    if (buf[1] == 'W') {
        if (buf[2] == 'N') {
            if (buf[3] == 'I') {
                if (buf[4] == 'T') {
                    printf("Found it!\n");
                    ((void(*)(0x0))0x0)();
                }
            }
        }
    }
}
```
What is Fuzzing?

```c
/* read file */
int n = read(buf, BUFSIZE);
if (n < 0)
    error("ERROR in read");

if (buf[0] == 'P') {
    if (buf[1] == 'W') {
        if (buf[2] == 'N') {
            if (buf[3] == 'I') {
                if (buf[4] == 'T') {
                    printf("Found it!\n");
                    ((void(*))0x0)();
                }
            }
        }
    }
}
```
What is Fuzzing?

/* read file */
int n = read(buf, BUFSIZE);
if (n < 0)
   error("ERROR in read");
   if (buf[0] == 'P') {
      if (buf[1] == 'W') {
         if (buf[2] == 'N') {
            if (buf[3] == 'I') {
               if (buf[4] == 'T') {
                  printf("Found it!\n");
               }
            }
         }
      }
   }
   return 0;}
What is Coverage-Guided Fuzzing?

```c
/* read file */
int n = read(buf, BUFSIZE);
if (n < 0)
    error("ERROR in read");

if (buf[0] == 'P') {
    if (buf[1] == 'W') {
        if (buf[2] == 'N') {
            if (buf[3] == 'I') {
                if (buf[4] == 'T') {
                    printf("Found it!\n");
                    ((void(*)())0x0)();
                }
            }
        }
    }
}
```
What is Coverage-Guided Fuzzing?

```c
/* read file */
int n = read(buf, BUFSIZE);
if (n < 0)
    error("ERROR in read");
if (buf[0] == 'P') {
    if (buf[1] == 'W') {
        if (buf[2] == 'N') {
            if (buf[3] == 'I') {
                if (buf[4] == 'T') {
                    printf("Found it!\n");
                    ((void(*)(0x0))0x0)(0);
                }
            }
        }
    }
}
```
What is Coverage-Guided Fuzzing?

```c
/* read file */
int n = read(buf, BUFSIZE);
if (n < 0)
  error("ERROR in read");
if (buf[0] == 'P') {
  if (buf[1] == 'W') {
    if (buf[2] == 'N') {
      if (buf[3] == 'I') {
        if (buf[4] == 'T') {
          printf("Found it!\n");
          ((void(*)())0x0)();
        }
      }
    }
  }
}
```
What is Coverage-Guided Fuzzing?

```c
/* read file */
int n = read(buf, BUFSIZE);
if (n < 0)
    error("ERROR in read");
if (buf[0] == 'P') {
    if (buf[1] == 'W') {
        if (buf[2] == 'N') {
            if (buf[3] == 'I') {
                if (buf[4] == 'T') {
                    printf("Found it!\n");
                    ((void(*)(void))0x0)(());
                }
            }
        }
    }
}
```
What is Coverage-Guided Fuzzing?

```c
/* read file */
int n = read(buf, BUFSIZE);
if (n < 0)
    error("ERROR in read");
if (buf[0] == 'P') {
    if (buf[1] == 'W') {
        if (buf[2] == 'N') {
            if (buf[3] == 'I') {
                if (buf[4] == 'T') {
                    printf("Found it!\n");
                    ((void(*)())0x0)();
                }
            }
        }
    }
}
```
What is Coverage-Guided Fuzzing?

```c
/* read file */
int n = read(buf, BUFSIZE);
if (n < 0)
    error("ERROR in read");
if (buf[0] == 'P') {
    if (buf[1] == 'W') {
        if (buf[2] == 'N') {
            if (buf[3] == 'I') {
                if (buf[4] == 'T') {
                    printf("Found it!\n");
                    ((void(*)(0x0))0x0)();
                }
            }
        }
    }
}
```
What is Coverage-Guided Fuzzing?

```c
/* read file */
n = read(buf, BUFSIZE);
if (n < 0)
    error("ERROR in read");

if (buf[0] == 'P') {
    if (buf[1] == 'W') {
        if (buf[2] == 'N') {
            if (buf[3] == 'I') {
                if (buf[4] == 'T') {
                    printf("Found it!\n");
                    ((void(*))0x0)();
                }
            }
        }
    }
}
```
What is Coverage-Guided Fuzzing?

```c
/* read file */
int n = read(buf, BUFSIZE);
if (n < 0) {
    error("ERROR in read");
}
if (buf[0] == 'P') {
    if (buf[1] == 'W') {
        if (buf[2] == 'N') {
            if (buf[3] == 'I') {
                if (buf[4] == 'T') {
                    printf("Found it!\n");
                    ((void(*)(void))0x0)(0);
                }
            }
        }
    }
}
```
What is Coverage-Guided Fuzzing?

```c
/* read file */

n = read(buf, BUFSIZE);
if (n < 0)
    error("ERROR in read");

if (buf[0] == 'P') {
    if (buf[1] == 'W') {
        if (buf[2] == 'N') {
            if (buf[3] == 'I') {
                if (buf[4] == 'T') {
                    printf("Found it!\n");
                    ((void(*)())0x0)();
                }
            }
        }
    }
}
```
What is Coverage-Guided Fuzzing?

```c
/* read file */
n = read(buf, BUFSIZE);
if (n < 0)
    error("ERROR in read");

if (buf[0] == 'P') {
    if (buf[1] == 'W') {
        if (buf[2] == 'N') {
            if (buf[3] == 'I') {
                if (buf[4] == 'T') {
                    printf("Found it!\n");
                    ((void(*)())0x0)();
                }
            }
        }
    }
}
```
What is Coverage-Guided Fuzzing?

```c
/* read file */

n = read(buf, BUFSIZE);
if (n < 0)
    error("ERROR in read");

if (buf[0] == 'P') {
    if (buf[1] == 'W') {
        if (buf[2] == 'N') {
            if (buf[3] == 'I') {
                if (buf[4] == 'T') {
                    printf("Found it!\n");
                    ((void(*)())0x0)();
                }
            }
        }
    }
}
```
What is Coverage-Guided Fuzzing?

```c
/* read file */
n = read(buf, BUFSIZE);
if (n < 0)
    error("ERROR in read");

if (buf[0] == 'P') {
    if (buf[1] == 'W') {
        if (buf[2] == 'N') {
            if (buf[3] == 'I') {
                if (buf[4] == 'T') {
                    printf("Found it!\n");
                    ((void(*)(void)0x0))();
                }
            }
        }
    }
}
```
What is Coverage-Guided Fuzzing?

```c
/* read file */
int n = read(buf, BUFSIZE);
if (n < 0)
    error("ERROR in read");
if (buf[0] == 'P') {
    if (buf[1] == 'W') {
        if (buf[2] == 'N') {
            if (buf[3] == 'I') {
                if (buf[4] == 'T') {
                    printf("Found it!\n");
                    ((void(*))0x0)();
                }
            }
        }
    }
}
```
What is Coverage-Guided Fuzzing?

```c
/* read file */
int n = read(buf, BUFSIZE);
if (n < 0)
    error("ERROR in read");
if (buf[0] == 'P') {
    if (buf[1] == 'W') {
        if (buf[2] == 'N') {
            if (buf[3] == 'I') {
                if (buf[4] == 'T') {
                    printf("Found it!\n");
                    ((void(*)())0x0)();
                }
            }
        }
    }
}
```
What is Coverage-Guided Fuzzing?

```c
/* read file */
int n = read(buf, BUFSIZE);
if (n < 0)
    error("ERROR in read");

if (buf[0] == 'P') {
    if (buf[1] == 'W') {
        if (buf[2] == 'N') {
            if (buf[3] == 'I') {
                if (buf[4] == 'T') {
                    printf("Found it!\n");
                    ((void(*)())0x0)();
                }
            }
        }
    }
}
```
What is Coverage-Guided Fuzzing?

```c
/* read file */
n = read(buf, BUFSIZE);
if (n < 0)
    error("ERROR in read");

if (buf[0] == 'P') {
    if (buf[1] == 'W') {
        if (buf[2] == 'N') {
            if (buf[3] == 'I') {
                if (buf[4] == 'T') {
                    printf("Found it!\n");
                    ((void(*)(void)0x0))();
                }
            }
        }
    }
}
```
What is Coverage-Guided Fuzzing?

```c
/* read file */

n = read(buf, BUFSIZE);
if (n < 0)
    error("ERROR in read");

if (buf[0] == 'P') {
    if (buf[1] == 'W') {
        if (buf[2] == 'N') {
            if (buf[3] == 'I') {
                if (buf[4] == 'T') {
                    printf("Found it!\n");
                    ((void(*)(void))0x0)();
                }
            }
        }
    }
}
```
American Fuzzy Lop aka AFL

<table>
<thead>
<tr>
<th>process timing</th>
<th>overall results</th>
</tr>
</thead>
<tbody>
<tr>
<td>run time</td>
<td>cycles done : 0</td>
</tr>
<tr>
<td></td>
<td>total paths : 30</td>
</tr>
<tr>
<td>last new path</td>
<td>uniq crashes : 1</td>
</tr>
<tr>
<td></td>
<td>uniq hangs : 0</td>
</tr>
<tr>
<td>last uniq crash</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>last uniq hang</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cycle progress</th>
<th>map coverage</th>
<th>path geometry</th>
</tr>
</thead>
<tbody>
<tr>
<td>now processing</td>
<td>map density</td>
<td>levels : 5</td>
</tr>
<tr>
<td></td>
<td>count coverage</td>
<td>pending : 20</td>
</tr>
<tr>
<td></td>
<td>findings in depth</td>
<td>pend fav : 8</td>
</tr>
<tr>
<td></td>
<td>favored paths</td>
<td>own finds : 29</td>
</tr>
<tr>
<td></td>
<td>new edges on</td>
<td>imported : n/a</td>
</tr>
<tr>
<td></td>
<td>total crashes</td>
<td>stability : 100.00%</td>
</tr>
<tr>
<td></td>
<td>total tmouts</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>stage progress</th>
<th>fuzzing strategy yields</th>
</tr>
</thead>
<tbody>
<tr>
<td>now trying :</td>
<td>bit flips : 6/680, 1/669, 2/647</td>
</tr>
<tr>
<td>arith 32/8</td>
<td>byte flips : 1/85, 0/74, 0/52</td>
</tr>
<tr>
<td>stage execs :</td>
<td>arithmetics : 1/4758, 0/3641, 0/730</td>
</tr>
<tr>
<td>0/545 (0.00%)</td>
<td>known ints : 0/282, 2/1351, 0/1893</td>
</tr>
<tr>
<td>total execs :</td>
<td>dictionary : 0/0, 0/0, 0/0</td>
</tr>
<tr>
<td>91.7k</td>
<td>havoc : 17/76.5k, 0/0</td>
</tr>
<tr>
<td>exec speed : 620.6/sec</td>
<td>trim : 12.77%/19, 0.00%</td>
</tr>
</tbody>
</table>

https://habr.com/ru/company/dsec/blog/449134/
SUMMARY: AddressSanitizer: heap-use-after-free ../../../..../C/LzmaDec.c:980 LzmaDec_FreeProbs

Shadow bytes around the buggy address:
0x0c1e7ffe9a70: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c1e7ffe9a80: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c1e7ffe9a90: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c1e7ffe9a90: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c1e7ffe9a90: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
0x0c1e7ffe9a90: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa

=>0x0c1e7ffe9ac0: fd [fd] fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd

Shadow byte legend (one shadow byte represents 8 application bytes):
Addressable: 00
A problem has been detected and Windows has been shut down to prevent further damage to your computer. If this is the first time you've seen this stop error:

- Modify your system code or a critical data structure was detected.
- Click "Send to Apple" to submit the report to Apple. This information is collected anonymously.

### Kernel AFL

<table>
<thead>
<tr>
<th>Runtime: 0:00:00:00:13</th>
<th>Performance: [ ]</th>
<th>876 t/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last Path: 00:00:00:08:09</td>
<td>Bittop: 03:22:00/00:00</td>
<td></td>
</tr>
<tr>
<td>Blacklisted: 0/0</td>
<td>Fuzzing: [ ]</td>
<td>2.5K</td>
</tr>
<tr>
<td>Cycles: 0</td>
<td>Blittopping: [ ]</td>
<td>30K</td>
</tr>
<tr>
<td>Level: 1/1</td>
<td>Arithmetic: [ ]</td>
<td>4.9K</td>
</tr>
<tr>
<td>Ping: 100%</td>
<td>Interesting: [ ]</td>
<td>4.9K</td>
</tr>
<tr>
<td>Pending: 1/1</td>
<td>Splicing: [ ]</td>
<td>4.9K</td>
</tr>
<tr>
<td>Skipped: 1/1</td>
<td>Total: 876 t/s</td>
<td></td>
</tr>
</tbody>
</table>

### MachineView

- CPU: 27.2%
- RAM: 45.1%
- HDD: 0%

### Stack

```
[ 290.719853] Stack: c07calic0 00000000 c07ca130 c17ca240 c07ca180 c17ca17
[ 290.720109] c180 c01496c9
[ 290.720180] [290.720364] [290.720468] [290.720626]
[ 290.720924] 003d fffffff1b c014f655 00000000 c09c120 52576e40
```

### Call Trace

```
[290.720626] Call Trace:
[290.720699] [290.720780] [290.720668] [290.720942]
```

### Dependencies

- com.apple.driver.ACM28.23-M
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
- com.apple.driver.ACM28.23.3
- com.apple.driver.ACM29.5
## Most Popular Languages in July 2019

<table>
<thead>
<tr>
<th>Jul 2019</th>
<th>Jul 2018</th>
<th>Change</th>
<th>Programming Language</th>
<th>Ratings</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>Java</td>
<td>15.058%</td>
<td>-1.08%</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td></td>
<td>C</td>
<td>14.211%</td>
<td>-0.45%</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>↑</td>
<td>Python</td>
<td>9.260%</td>
<td>+2.90%</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>↓</td>
<td>C++</td>
<td>6.705%</td>
<td>-0.91%</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>↑</td>
<td>C#</td>
<td>4.365%</td>
<td>+0.57%</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>↓</td>
<td>Visual Basic .NET</td>
<td>4.208%</td>
<td>-0.04%</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>↑</td>
<td>JavaScript</td>
<td>2.304%</td>
<td>-0.53%</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>↓</td>
<td>PHP</td>
<td>2.167%</td>
<td>-0.67%</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td></td>
<td>SQL</td>
<td>1.977%</td>
<td>-0.36%</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td></td>
<td>Objective-C</td>
<td>1.686%</td>
<td>+0.23%</td>
</tr>
</tbody>
</table>

[https://www.tiobe.com/tiobe-index/]
Fuzzing is Very Hot Today!
OSS-Fuzz Project

- ~160 open-source projects
- ~half-trillion test cases per week

Open Issues Count per Month
Downsides. Volatile Paths

AAAAAAAAAAA ➔

[Diagram with various nodes and connections, some highlighted in green]
Downsides. Volatile Paths

AAAAAAA

ABAAAAAA
Downsides. Volatile Paths

AAAAAA

ABAAAA

ABAAAAA
Downsides. Volatile Paths

AAAAAAAAAAA

ABAAAAAAA

ABAAAAAAA
Downsides. Volatile Paths
# Downsides. Volatile Paths

---

```plaintext
american fuzzy lop 2.52b (7z)

<table>
<thead>
<tr>
<th>process timing</th>
<th>overall results</th>
</tr>
</thead>
<tbody>
<tr>
<td>run time</td>
<td>cycles done: 0</td>
</tr>
<tr>
<td>last new path</td>
<td>total paths: 203</td>
</tr>
<tr>
<td>last uniq crash</td>
<td>uniq crashes: 0</td>
</tr>
<tr>
<td>last uniq hang</td>
<td>uniq hangs: 0</td>
</tr>
<tr>
<td>cycle progress</td>
<td></td>
</tr>
<tr>
<td>now processing</td>
<td></td>
</tr>
<tr>
<td>paths timed out</td>
<td></td>
</tr>
<tr>
<td>stage progress</td>
<td></td>
</tr>
<tr>
<td>now trying</td>
<td></td>
</tr>
<tr>
<td>stage execs</td>
<td></td>
</tr>
<tr>
<td>total execs</td>
<td></td>
</tr>
<tr>
<td>exec speed</td>
<td></td>
</tr>
<tr>
<td>fuzzing strategy yields</td>
<td></td>
</tr>
</tbody>
</table>

| bit flips                    | favored paths: 106 (52.22%) |
| byte flips                   | new edges on: 132 (65.02%) |
| arithmetics                  | total crashes: 0 (0 unique) |
| known ints                   | total timeouts: 0 (0 unique) |
| dictionary                   |                            |
| havoc                        |                            |
| trim                         |                            |

stability: 44.90%
```
Downsides. Parallelization algorithm

- Parallelization is an obvious solution to speed up fuzzing and find more bugs.
- AFL was not designed to be parallel fuzzer

AFL master folder    AFL slave #1    AFL slave #2
Downsides. Parallelization algorithm

- Parallelization is an obvious solution to speed up fuzzing and find more bugs.
- AFL was not designed to be parallel fuzzer
Downsides. Parallelization algorithm

- Parallelization is an obvious solution to speed up fuzzing and find more bugs.
- AFL was not designed to be parallel fuzzer

AFL master folder  AFL slave #1  AFL slave #2
Downsides. Parallelization algorithm

- Parallelization is an obvious solution to speed up fuzzing and find more bugs.
- AFL was not designed to be parallel fuzzer
Downsides. Parallelization algorithm

- Parallelization is an obvious solution to speed up fuzzing and find more bugs.
- AFL was not designed to be parallel fuzzer
Network apps fuzzing. Current situation

- **Linux:**
  - AFL’s forks, honggfuzz and blind fuzzers

- **Windows**
  - winAFL network mode

- **OS X**
  - honggfuzz?
Windows applications fuzzing

winAFL
clang (libfuzzer/honggfuzz)
OS X applications fuzzing

- Source code is required. Target should be able to compile with clang
- DynamoRIO has no official support of OS X
- Intel PIN has partial OS X support
Some Related Works & Tools

- The author is not the first one who wants to improve AFL.
  - Userland: AFLSmart, AFLFast, winAFL, libfuzzer, driller, QSYM and others.
  - Kernel: syzkaller, kAFL, TriforceAFL and others.

- Systematic research on all existing fuzzers:

- Some Presentations at DEF CON/BlackHat:
### State-of-the-art Userland Fuzzers

<table>
<thead>
<tr>
<th></th>
<th>AFL winAFL</th>
<th>HongFuzz</th>
<th>libFuzzer</th>
<th>Desired fuzzer</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Network fuzzing</strong></td>
<td>No (Unix)</td>
<td>Yes</td>
<td>No</td>
<td>Yes (all platforms)</td>
</tr>
<tr>
<td></td>
<td>Yes (Windows)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Volatile Paths</strong></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Multiple Mutation Strategies</strong></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Share over network</strong></td>
<td>Partial</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Supported Platform</strong></td>
<td>Linux Windows</td>
<td>Open/NetBSD GNU/Linux Windows/Cygwin Android OS X</td>
<td>Anywhere where LLVM exist</td>
<td>Anywhere where Python exist</td>
</tr>
<tr>
<td><strong>Language</strong></td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>Python</td>
</tr>
</tbody>
</table>
Manul Overview

- Manul - an open-source fuzzer written in pure Python.
  - Easy-to-use, pull and run concept.
  - Coverage-guided fuzzing using AFL-GCC or DBI (Intel Pin or DynamoRIO).
  - Parallel fuzzing is a basic feature.
  - Default mutators.
  - Third-party data mutators (Radamsa + AFL currently supported).
  - Network fuzzing is supported by default.
  - Blackbox binaries fuzzing.
  - Supported: Linux, MacOS (beta) and Windows or any other OS where Python exist.
Why Manul?

Pallas’s Cat (lat. Otocolobus Manul)
Manul Architecture

- Mutators (plugins)
- User Interface
- Manul Network Module
- Core Module
- Code Coverage Analysis Module
- Target
  - Instrumentation module
- Fuzzer
- Shared Memory
Volatile Paths Detection
Volatile Paths Detection

ABAAAAAAAAA Run & Calibrate
Volatile Paths Detection

Parallel fuzzing. Python Multiprocessing
Parallel fuzzing. Python Multiprocessing

Instance #1

Instance #2

Instance #3

Main Process

Corpus:
Parallel fuzzing. Python Multiprocessing

Instance #1
Instance #2
Instance #3

Main Process

Corpus:
Parallel fuzzing. Python Multiprocessing

Instance #1  Instance #2  Instance #3

Main Process

Corpus:
Parallel fuzzing. Python Multiprocessing
Parallel fuzzing. Python Multiprocessing
Main Process

Instance #1

Instance #2

Instance #3

Remote Instance (Main Process)

Target

Target

Target

Remote Instance #1

Remote Instance #2

Global shared memory

Main Process

SHM

SHM

SHM

SHM

SHM

Bitmap synchronization over network
Third Party Mutators

- AFL strategy (ported to Python) and Radamsa (as a shared library)

Custom Python Mutator:

- `def init(fuzzer_id)`
- `def mutate(data_to_mutate)`
Network Application Fuzzing (Experimental)

Client mode

TCP|UDP

Manul ———> Target

Server mode

Connect

Test case
(TCP|UDP)

Manul ———> Target
Blackbox Binaries Fuzzing

Windows: DynamoRIO: ~x30 overhead
Linux: Intel Pin: ~x45 overhead
DynamoRIO: ~x20 overhead
<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Time</th>
<th>Python x64</th>
<th>Notes</th>
<th>Other Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmd.exe</td>
<td>0.13</td>
<td>7916 K</td>
<td>10.69</td>
<td>16364 K Python</td>
<td>Python Software Foundation</td>
</tr>
<tr>
<td>cmd.exe</td>
<td>0.45</td>
<td>3376 K</td>
<td>3.476</td>
<td>16364 K Python</td>
<td>Python Software Foundation</td>
</tr>
<tr>
<td>cmd.exe</td>
<td>0.46</td>
<td>1084 K</td>
<td>2.984</td>
<td>16364 K Python</td>
<td>Python Software Foundation</td>
</tr>
<tr>
<td>cmd.exe</td>
<td>0.43</td>
<td>344 K</td>
<td>1.204</td>
<td>16364 K Python</td>
<td>Python Software Foundation</td>
</tr>
<tr>
<td>cmd.exe</td>
<td>0.38</td>
<td>3384 K</td>
<td>3.436</td>
<td>16364 K Python</td>
<td>Python Software Foundation</td>
</tr>
<tr>
<td>cmd.exe</td>
<td>0.33</td>
<td>1088 K</td>
<td>2.984</td>
<td>16364 K Python</td>
<td>Python Software Foundation</td>
</tr>
<tr>
<td>cmd.exe</td>
<td>0.37</td>
<td>3476 K</td>
<td>3.436</td>
<td>16364 K Python</td>
<td>Python Software Foundation</td>
</tr>
<tr>
<td>cmd.exe</td>
<td>0.41</td>
<td>1084 K</td>
<td>3.028</td>
<td>16364 K Python</td>
<td>Python Software Foundation</td>
</tr>
<tr>
<td>cmd.exe</td>
<td>0.34</td>
<td>344 K</td>
<td>3.436</td>
<td>16364 K Python</td>
<td>Python Software Foundation</td>
</tr>
<tr>
<td>cmd.exe</td>
<td>0.44</td>
<td>1092 K</td>
<td>3.028</td>
<td>16364 K Python</td>
<td>Python Software Foundation</td>
</tr>
<tr>
<td>cmd.exe</td>
<td>0.49</td>
<td>3376 K</td>
<td>3.436</td>
<td>16364 K Python</td>
<td>Python Software Foundation</td>
</tr>
<tr>
<td>cmd.exe</td>
<td>0.41</td>
<td>1052 K</td>
<td>3.028</td>
<td>16364 K Python</td>
<td>Python Software Foundation</td>
</tr>
<tr>
<td>cmd.exe</td>
<td>0.16</td>
<td>7608 K</td>
<td>10.69</td>
<td>16364 K Python</td>
<td>Python Software Foundation</td>
</tr>
<tr>
<td>cmd.exe</td>
<td>0.41</td>
<td>3372 K</td>
<td>3.472</td>
<td>16364 K Python</td>
<td>Python Software Foundation</td>
</tr>
<tr>
<td>cmd.exe</td>
<td>0.42</td>
<td>1092 K</td>
<td>3.028</td>
<td>16364 K Python</td>
<td>Python Software Foundation</td>
</tr>
<tr>
<td>cmd.exe</td>
<td>0.20</td>
<td>7608 K</td>
<td>10.69</td>
<td>16364 K Python</td>
<td>Python Software Foundation</td>
</tr>
</tbody>
</table>
### Interface & Logo

**Manul v 0.1. All fuzzers summary**

<table>
<thead>
<tr>
<th>Mode</th>
<th>Strategy</th>
<th>Loggl</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBI</td>
<td>radamsa</td>
<td></td>
</tr>
</tbody>
</table>

#### Timing
- Time: 0d 2h 58m 10s
- Last new crash found: 0d 1h 34m 10s
- Last new path found: 0d 0h 11m 53s

#### Coverage statistics
- Volatile bytes: 0
- Bitmap coverage: 21.52%
- New paths found: 36

#### Results
- **Crashes:** 1
- Unique crashes: 0
- Exceptions: 17473

#### Performance
- Exec/sec: 2.72667
- Iterations: 5759
- Files in queue: 39
Command Line Arguments

Manul - coverage-guided parallel fuzzing for native applications.

Positional arguments:
- target_binary  The target binary and options to be executed.

Optional arguments:
- -h, --help    show this help message and exit
- -n NFUZZERS   Number of parallel fuzzers
- -s            Run dumb fuzzing (no code instrumentation)
- -c CONFIG     Path to config file with additional options (see manul.config)
- -r            Restore previous session

Required parameters:
- -i INPUT     Path to directory with initial corpus
- -o OUTPUT    Path to output directory
DEMO

(Manul)
Case Study I. Poppler

- Poppler is an open-source library for rendering PDF documents on GNU/Linux
  - Millions of users across the world. Default package on Ubuntu
  - Integrated with Evince, LibreOffice, Inkscape and many other applications
- Written in C++
- Participate in OSS-Fuzz program (tough target)
Case Study I. Poppler. Fuzzing Setup

- 491 PDF files (same corpus used by OSS-Fuzz)
- 24 hours, 78 parallel jobs
- AFL ver. 2.52b & Manul ver. 0.2
- Intel Xeon CPU E5-2698 v4 @2.20GHz 1TB RAM
Case Study I. Execution Speed
Case Study I. Paths Found
Case Study I. Why Manul outperformed AFL

- Manul corpus parallelization algorithm demonstrates better performance on large targets
- Radamsa + AFL is better than only AFL
- Volatile paths suppression seems to work
Case Study I. Manul Findings

CVE-2019-9631. 9.8 Critical. Poppler 0.74.0 has a heap-based buffer over-read in the CairoRescaleBox.cc downsample_row_box_filter function.

CVE-2019-7310. 8.8 High. Poppler 0.74.0. A heap-based buffer over-read (due to an integer signedness error in the XRef::getEntry function in XRef.cc) allows remote attackers to cause a denial of service (application crash) or possibly have unspecified other impact via a crafted PDF document, as demonstrated by pdftocairo.

CVE-2019-9959 (X.X. High) In Poppler (latest), JPXStream::init doesn’t have a check for negative values of stream length thereby making it possible to allocate large memory chunk on heap with size controlled by an attacker.

Non-security related:
1. Division by zero in CairoRescalBox::downScaleImage
2. Null-pointer dereference in ExtGState
3. Stack-overflow (recursion) in libcairo
static void downsample_row_box_filter (int start, int width, int *src, int *dest, int coverage[], int pixel_coverage)
{
    //-----truncated------>
    while (x < start + width)
    {
        int box = 1 << FIXED_SHIFT;
        int start_coverage = coverage[x];
        a = (*(src >> 24) & 0xff) * start_coverage;
        r = (*(src >> 16) & 0xff) * start_coverage;
        g = (*(src >>  8) & 0xff) * start_coverage;
        b = (*(src >>  0) & 0xff) * start_coverage;
        src++;
        x++;
        box <<= start_coverage;

        while (box > pixel_coverage)
        {
            a += (*(src >> 24) & 0xff) * pixel_coverage; // <---- overrun happens here
            r += (*(src >> 16) & 0xff) * pixel_coverage;
            g += (*(src >>  8) & 0xff) * pixel_coverage;
            b += (*(src >>  0) & 0xff) * pixel_coverage;
            src++;
            box -= pixel_coverage;
        }

        if (box > 0)
        {
            a = (*(src >> 24) & 0xff) * box;
            r = (*(src >> 16) & 0xff) * box;
            g = (*(src >>  8) & 0xff) * box;
            b = (*(src >>  0) & 0xff) * box;
        }

        a >>= FIXED_SHIFT;
        r >>= FIXED_SHIFT;
        g >>= FIXED_SHIFT;
        b >>= FIXED_SHIFT;

        *dest = (a << 24) | (r << 16) | (g << 8) | b;
        dest++;
    }
}
Case Study I. Poppler. CVE 2019-9631
Case Study I. Poppler. CVE 2019-9959

```cpp
void JPXStream::init()
{
    Object oLen, cspace, smaskInData;
    if (getDict()) {
        oLen = getDict()->lookup("Length");
        cspace = getDict()->lookup("ColorSpace");
        smaskInData = getDict()->lookup("SMaskInData");
    }

    int bufSize = BUFFER_INITIAL_SIZE;
    if (oLen.isInt()) bufSize = oLen.getInt();

    bool indexed = false;
    if (cspace.isArray() && cspace.arrayGetLength() > 0) {
        const Object cstype = cspace.arrayGet(0);
        if (cstype.isName("Indexed")) indexed = true;
    }

    priv->smaskInData = 0;
    if (smaskInData.isInt()) priv->smaskInData = smaskInData.getInt();

    int length = 0;
    unsigned char *buf = str->toUnsignedChars(&length, bufSize);
    priv->init2(OPJ_CODEC JP2, buf, length, indexed);
    gfree(buf);
```
```c
inline unsigned char *toUnsignedChars(int *length, int initialSize = 4096, int sizeIncrement = 4096) {
    int readChars;
    unsigned char *buf = (unsigned char *)malloc(initialSize);
    int size = initialSize;
    *length = 0;
    int charsToRead = initialSize;
    bool continueReading = true;
    reset();
    while (continueReading && (readChars = doGetChars(charsToRead, &buf[*length])) != 0) {
        *length += readChars;
        if (readChars == charsToRead) {
            if (lookChar() != EOF) {
                size += sizeIncrement;
                charsToRead = sizeIncrement;
                buf = (unsigned char *)realloc(buf, size);
            } else {
                continueReading = false;
            }
        } else {
            continueReading = false;
        }
    }
    return buf;
}
```
Case Study I. Poppler. CVE 2019-7310

```c
XRefEntry *XRef::getXRefEntry(int i, bool complainIfMissing)
{
    if (i >= size || entries[i].type == xrefEntryNone)
        return &dummyXRefEntry;

    if (!xrefStream) {
        if (unlikely(i >= capacity)) {
            error(errInternal, -1, "Request for out-of-bounds XRef entry [[0:d]]", i);
            return &dummyXRefEntry;
        }
        if (!parseEntry(mainXRrefEntriesOffset + 26*i, &entries[i])) {
            error(errSyntaxError, -1, "Failed to parse XRef entry [[0:d]].", i);
        }
    } else {
        // Read XRef tables until the entry we're looking for is found
        readXRefUntil(i);

        // We might have reconstructed the xref
        // Check again i is in bounds
        if (unlikely(i >= size)) {
            return &dummyXRefEntry;
        }

        if (entries[i].type == xrefEntryNone) {
            if (complainIfMissing) {
                error(errSyntaxError, -1, "Invalid XRef entry \(0:d\).", i);
            }
            entries[i].type = xrefEntryFree;
        }
    }
}
```

return &entries[i];
Case Study I. Poppler. CVE 2019-7310

Stream *Parser::makeStream(Object **dict, unsigned char *fileKey,
   CryptAlgorithm encAlgorithm, int keyLength,
   int objNum, int objGen, int recursion,
   bool strict) {

   BaseStream *baseStr;
   Stream *str;
   Goffset length;
   Goffset pos, endPos;

   if (xref) {
      XRefEntry *entry = xref->getEntry(objNum, false);
      if (entry) {
         if (entry->getFlag(XRefEntry::Parsing) ||
            (objNum == 0 && objGen == 0)) {
            entry->setFlag(XRefEntry::Parsing, true);
         } else {
            error(errSyntaxError, getPos(),
               "Object '{0:d} {1:d} obj' is being already parsed", objNum, objGen);
            return nullptr;
         }
      }
   }
}
Case Study II. Zeek IDS

- Zeek (former Bro) is a world’s most powerful open-source network analysis framework
  - Thousand of companies use Zeek as IDS
  - JA3 plugin for Zeek is a very powerful tool to detect suspicious connections of malware with C2
- BroCon happens in Arlington, VA every October
- Written in C++, very high-quality code, fuzzing was done using libfuzzer by development team in the past
Zeek Fuzzing Wrapper Example

```
ssha = new analyzer::SSH::SSH_Analyzer(conn);
ssha->SetTCP(tcpa);
ssha->DeliverStream(strlen("SSH-2.0-Cisco-1.25\n") + 1, ssh_server_name, false); /* server's protocol */
ssha->DeliverStream(strlen("SSH-2.0-Cisco-1.25\n") + 1, ssh_client_name, true);  /* client protocol */
ssha->DeliverStream(DataSize, Data, false); /* false - from server to client */
ssha->Done();
free(ssh_server_name);
free(ssh_client_name);
delete ssha;
```

- Implemented for HTTP, IRC, KRB, DNP3, SSH, DNS, ICMP, LOGIN, FTP, IMAP
Case Study II. Findings

CVE-2018-17019 (7.5. High). In Zeek IDS through 2.5.5, there is a DoS in IRC protocol names command parsing in analyzer/protocol/irc/IRC.cc

CVE-2018-16807 (7.5. High). In Zeek IDS through 2.5.5, there is a memory leak potentially leading to DoS in scripts/base/protocols/krb/main.bro in the Kerberos protocol parser.

CVE-2019-12175. (XX High). In Zeek IDS, there is a DoS in Kerberos protocol parser in analyzer/protocol/krb/KRB.cc
CVE-2018-16807

#1 0x16d0f10 in binpac::KRB_TCP::proc_krb_kdc_req_arguments(binpac::KRB_TCP::KRB_KDC_REQ*, analyzer::Analyzer*)
#2 0x16d0994 in binpac::KRB_TCP::KRB_Conn::proc_krb_kdc_req_msg(binpac::KRB_TCP::KRB_KDC_REQ*)
#3 0x16f6038 in binpac::KRB_TCP::KRB_AS_REQ::Parse(unsigned char const*, unsigned char const*, binpac::KRB_TCP::ContextKRB_TCP*, int)
IRC Protocol

1. NICK amy
2. USER amy * * :Amy Pond
3. :bar.example.com 001 amy :Welcome to the Internet Relay Network amy!amy@foo.example.com
CVE 2018-16807. Packet Example

Send packet that contains: “353 “ on IRC port 6666
CVE-2019-12175

==103310==ERROR: AddressSanitizer: SEGV on unknown address 0x000000000000 (pc 0x55a797d15b75 bp 0x7ffe14590cb0 sp 0x7ffe14590330 T0)
#0 0x55a797d15b74 in binpac::KRB_TCP::proc_padata(binpac::KRB_TCP::KRB_PA_Data_Sequence const*, analyzer::Analyzer*, bool)
#1 0x55a797d3d36a in binpac::KRB_TCP::proc_krb_kdc_req_arguments(binpac::KRB_TCP::KRB_KDC_REQ*, analyzer::Analyzer*)
#2 0x55a797d3f61b in binpac::KRB_TCP::KRB_Conn::proc_krb_kdc_req_msg(binpac::KRB_TCP::KRB_KDC_REQ*)
#3 0x55a797d65032 in binpac::KRB_TCP::KRB_AS_REQ::Parse(unsigned char const*, unsigned char const*, binpac::KRB_TCP::ContextKRB_TCP*, int)
#4 0x55a797d65032 in binpac::KRB_TCP::KRB_PDU::Parse(unsigned char const*, unsigned char const*, binpac::KRB_TCP::ContextKRB_TCP*)
#5 0x55a797d69717 in binpac::KRB_TCP::KRB_PDU_TCP::ParseBuffer(binpac::FlowBuffer*, binpac::KRB_TCP::ContextKRB_TCP*)
#6 0x55a797d69717 in binpac::KRB_TCP::KRB_Flow::NewData(unsigned char const*, unsigned char const*)
DEMO

(example of CVE 2019-12175 DoS in Zeek)
**List of Bugs Found**

<table>
<thead>
<tr>
<th>Bugs</th>
<th>Project</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2019-XXXX, CVE-2019-XXXX</td>
<td>7-Zip 19.00 for Windows</td>
</tr>
<tr>
<td>Awaiting assignment from MITRE and fix from maintainer</td>
<td></td>
</tr>
<tr>
<td>Awaiting assignment from MITRE and fix from maintainer</td>
<td></td>
</tr>
<tr>
<td>CVE-2019-XXXX, CVE-2019-XXXX</td>
<td>Unarchiver for MacOS</td>
</tr>
<tr>
<td>Awaiting assignment from MITRE and fix from maintainer</td>
<td></td>
</tr>
</tbody>
</table>
Discussion & Future Work

- AFL’s forker server is strongly required
- Add Intel PTrace support
- More mutation algorithms
  - + structure-aware fuzzing
- Better MacOS support
- Better network fuzzing support
- CLANG-based instrumentation
Conclusion

- Fuzzing is #1 technique for vulnerability research in memory-unsafe languages
- Manul is a fully functional tool for efficient coverage-guided fuzzing.
  - Multiple third-party mutators, volatile paths suppression, efficient parallelization algorithm, blackbox binaries fuzzing
- 13 new bugs in 4 widely-used open-source projects.
- Pull & try! [https://github.com/mxmssh/manul](https://github.com/mxmssh/manul)
  - pip install psutil & git clone [https://github.com/mxmssh/manul](https://github.com/mxmssh/manul)
Thank you!

https://github.com/mxmssh/manul
Twitter: https://twitter.com/MShudrak
Linkedin: https://www.linkedin.com/in/mshudrak/