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:
  - EAOS: Extremely Abstract Operating System for Malware Analysis (at IBM Research 2015-2017)
  - 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:

virus BULLETIN

phd Positive Hack Days

BSIDES SAN FRANCISCO

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?

AAAAA →

```
/* 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?

AAAAA →

```
/* 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?

BAAAA →

```
/* 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?

CAAAA →

```
/* 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?

PAAAA →

```
/* 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?

PWNIT

```
/* 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?

PWNIT

Very unlikely!

```
/* 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?

AAAAA →

```
/* 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?

AAAAA →

```
/* 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?

BAAAA →

```
/* 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?

CAAAA →

```
/* 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?

PAAAA →

```
/* 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?

Input queue

AAAAA
PAAAA

```
/* 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?

Input queue

AAAAA
PBAAA

```
/* 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?

Input queue

AAAAA
PCAAA

```
/* 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?

Input queue

AAAAA
PWAAA

```
/* 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?

Input queue

AAAAA
PAAAA
PWAAA

```
/* 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?

Input queue

AAAAA
PAAAA
PWBAA

```
/* 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?

Input queue

AAAAA
PAAAA
PWNAA

```
/* 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?

Input queue

AAAAA
PAAAA
PWNAA
PWNBA

```
/* 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?

Input queue

AAAAA
PAAAA
PWNAA
PWNIA

```
/* 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?

Input queue

AAAAA
PAAAA
PWNAA
PWNIA
PWNIB

```
/* 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?

Input queue

AAAAA
PAAAA
PWNAA
PWNIA
PWNIC

```
/* 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?

Input queue

AAAAA
PAAAA
PWNAA
PWNIA
PWNIT

```
/* 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)();
                }
            }
        }
    }
}
```

American Fuzzy Lop aka AFL

```
                american fuzzy lop 2.52b (handshake)

┌─ process timing ─────────────────────────────────────┬─ overall results ─────┐
│        run time : 0 days, 0 hrs, 2 min, 11 sec       │  cycles done : 0      │
│   last new path : 0 days, 0 hrs, 0 min, 1 sec        │  total paths : 30     │
│ last uniq crash : 0 days, 0 hrs, 0 min, 29 sec       │ uniq crashes : 1      │
│  last uniq hang : none seen yet                      │   uniq hangs : 0      │
├─ cycle progress ────────────────────┬─ map coverage ─┴───────────────────────┤
│  now processing : 19 (63.33%)       │    map density : 1.25% / 1.63%         │
│ paths timed out : 0 (0.00%)         │ count coverage : 1.36 bits/tuple       │
├─ stage progress ────────────────────┼─ findings in depth ────────────────────┤
│  now trying : arith 32/8            │ favored paths : 17 (56.67%)            │
│ stage execs : 0/545 (0.00%)         │  new edges on : 21 (70.00%)            │
│ total execs : 91.7k                 │ total crashes : 113 (1 unique)         │
│  exec speed : 620.6/sec             │  total tmouts : 0 (0 unique)           │
├─ fuzzing strategy yields ───────────┴───────────────┬─ path geometry ────────┤
│   bit flips : 6/680, 1/669, 2/647                   │    levels : 5          │
│  byte flips : 1/85, 0/74, 0/52                      │   pending : 20         │
│ arithmetics : 1/4758, 0/3641, 0/730                 │  pend fav : 8          │
│  known ints : 0/282, 2/1351, 0/1893                 │ own finds : 29         │
│  dictionary : 0/0, 0/0, 0/0                         │  imported : n/a        │
│       havoc : 17/76.5k, 0/0                         │ stability : 100.00%    │
│        trim : 12.77%/19, 0.00%                      ├────────────────────────┘
└─────────────────────────────────────────────────────┘          [cpu000: 27%]
```

https://habr.com/ru/company/dsec/blog/449134/

Command Prompt

```
==========
==9704==ERROR: AddressSanitizer: requested allocation size 0x10000000000008b6 (0x10000000000018b8 after adjustments for
alignment, red zones etc.) exceeds maximum supported size of 0x10000000000 (thread T0)
    #0 0x7ffd8e45df84 in malloc+0xc4 (C:\Users\7z\Desktop\7z1900_asan\CPP\7zip\UI\Console\x64\7z.dll+0x18045df84)
    #1 0x7ffd8e4865be in _ubsan_get_current_report_data+0x76e (C:\Users\7z\Desktop\7z1900_asan\CPP\7zip\UI\Console\x64\7
z.dll+0x1804865be)
    #2 0x7ffd8e102d0c in Ordinal0+0x102d0c (C:\Users\7z\Desktop\7z1900_asan\CPP\7zip\UI\Console\x64\7z.dll+0x180102d0c)
    #3 0x7ffd8e1dafd0 in _sanitizer_malloc_hook__dll+0x78c40 (C:\Users\7z\Desktop\7z1900_asan\CPP\7zip\UI\Console\x64\7z
.dll+0x1801dafd0)
    #4 0x7ff76fd2bd28 in Ordinal0+0x7bd28 (C:\Users\7z\Desktop\7z1900_asan\CPP\7zip\UI\Console\x64\7z.exe+0x14007bd28)
    #5 0x7ff76fd9d1c5 in _sanitizer_get_unmapped_bytes+0x558b5 (C:\Users\7z\Desktop\7z1900_asan\CPP\7zip\UI\Console\x64\
7z.exe+0x1400ed1c5)
    #6 0x7ff76fda5ba1 in _sanitizer_get_unmapped_bytes+0x5e291 (C:\Users\7z\Desktop\7z1900_asan\CPP\7zip\UI\Console\x64\
7z.exe+0x1400f5ba1)
    #7 0x7ff76fde076f in _ubsan_get_current_report_data+0xdcf (C:\Users\7z\Desktop\7z1900_asan\CPP\7zip\UI\Console\x64\7
z.exe+0x14013076f)
    #8 0x7ffdd01d7973 in BaseThreadInitThunk+0x13 (C:\Windows\System32\KERNEL32.DLL+0x180017973)
    #9 0x7ffdd0e5a270 in RtlUserThreadStart+0x20 (C:\Windows\SYSTEM32\ntdll.dll+0x18006a270)

==9704==HINT: if you don't care about these errors you may set allocator_may_return_null=1
SUMMARY: AddressSanitizer: allocation-size-too-big (C:\Users\7z\Desktop\7z1900_asan\CPP\7zip\UI\Console\x64\7z.dll+0x180
45df84) in malloc+0xc4
==9704==ABORTING

C:\Users\7z\Desktop\7z1900_asan\CPP\7zip\UI\Console\x64>
```

in_rar_crashes — -bash — 238×54

```
==24716==ERROR: AddressSanitizer: heap-buffer-overflow on address 0x611000006430 at pc 0x000108248a04 bp 0x7ffee7c838b0 sp 0x7ffee7c838a8
READ of size 4 at 0x611000006430 thread T0
    #0 0x108248a03 in Branch XADPrefixCode.m:22
    #1 0x10824d07c in LeftBranch XADPrefixCode.m:25
    #2 0x10824891a in IsLeafNode XADPrefixCode.m:38
    #3 0x10824c1f8 in MakeTable XADPrefixCode.m:344
    #4 0x10824c537 in MakeTable XADPrefixCode.m:366
    #5 0x10824c537 in MakeTable XADPrefixCode.m:366
    #6 0x10824c4cd in MakeTable XADPrefixCode.m:365
    #7 0x10824c4cd in MakeTable XADPrefixCode.m:365
    #8 0x10824c1b6 in -[XADPrefixCode _makeTable] XADPrefixCode.m:415
    #9 0x108248255 in CSInputNextSymbolUsingCode XADPrefixCode.m:51
    #10 0x1080c1427 in -[XADRAR30Handle expandToPosition:] XADRAR30Handle.m:247
    #11 0x1080c09c7 in -[XADRAR30Handle produceBlockAtOffset:] XADRAR30Handle.m:150
    #12 0x10826caf5 in -[XADBlockStreamHandle _readNextBlock] CSBlockStreamHandle.m:97
    #13 0x10826c6e6 in -[XADBlockStreamHandle streamAtMost:toBuffer:] CSBlockStreamHandle.m:82
    #14 0x108288210 in -[XADStreamHandle readAtMost:toBuffer:] CSStreamHandle.m:154
    #15 0x108289f73 in -[XADSubHandle readAtMost:toBuffer:] CSSubHandle.m:82
    #16 0x1082522d0 in -[XADCRCHandle streamAtMost:toBuffer:] XADCRCHandle.m:76
    #17 0x108288210 in -[XADStreamHandle readAtMost:toBuffer:] CSStreamHandle.m:154
    #18 0x108056ba2 in -[XADUnarchiver runExtractorWithDictionary:outputTarget:selector:argument:] XADUnarchiver.m:609
    #19 0x108056444 in -[XADUnarchiver runExtractorWithDictionary:outputHandle:] XADUnarchiver.m:554
    #20 0x1083d14a4 in -[XADSimpleUnarchiver unarchiver:shouldExtractEntryWithDictionary:suggestedPath:] XADSimpleUnarchiver.m:816
    #21 0x10805164d in -[XADUnarchiver extractEntryWithDictionary:as:forceDirectories:] XADUnarchiver.m:171
    #22 0x108050991 in -[XADUnarchiver extractEntryWithDictionary:] XADUnarchiver.m:129
    #23 0x1083cda68 in -[XADSimpleUnarchiver _unarchiveRegularArchive] XADSimpleUnarchiver.m:461
    #24 0x1083ccb65 in -[XADSimpleUnarchiver unarchive] XADSimpleUnarchiver.m:396
    #25 0x107ffe014 in main unar.m:267
    #26 0x7fff75e1a3d4 in start (libdyld.dylib:x86_64+0x163d4)

0x611000006430 is located 16 bytes to the left of 200-byte region [0x611000006440,0x611000006510)
allocated by thread T0 here:
    #0 0x1087763d7 in wrap_realloc (libclang_rt.asan_osx_dynamic.dylib:x86_64h+0x5e3d7)
```

```
SUMMARY: AddressSanitizer: heap-use-after-free ../../../../C/LzmaDec.c:980 LzmaDec_FreeProbs
Shadow bytes around the buggy address:
  0x0c1e7fff9a70: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c1e7fff9a80: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c1e7fff9a90: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c1e7fff9aa0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
  0x0c1e7fff9ab0: fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa fa
=>0x0c1e7fff9ac0: fd fd[fd]fd fd fd fd fd fd fd fd fd fd fd fd fd
  0x0c1e7fff9ad0: fd fd fd fd fd fa fa fa fa fa fa fa fa fa fa 00 00
  0x0c1e7fff9ae0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x0c1e7fff9af0: 00 00 00 04 fa fa fa fa fa fa fa fa fa fd fd fd fd
  0x0c1e7fff9b00: fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd fd
  0x0c1e7fff9b10: fd fa fa fa fa fa fa fa fa fa fa fa fd fd fd fd fd fd
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
```

Problem Report for macOS

Your computer was restarted because of a problem.

Click "Send to Apple" to submit the report to Apple. This information is collected anonymously.

▸ Comments

Problem Details and System Configuration

```
Anonymous UUID:        F3F14D81-896B-D228-C0AF-AA54A22C7368

Mon Feb 19 21:13:37 2018

*** Panic Report ***
panic(cpu 6 caller 0xffffff8008dff6f6): trying to interlock destroyed mutex (0xffffff80321b9d98)
Backtrace (CPU 6), Frame : Return Address
0xffffff9227933b60 : 0xffffff8008e4f606
0xffffff9227933bb0 : 0xffffff8008f7c654
0xffffff9227933bf0 : 0xffffff8008f6e149
0xffffff9227933c70 : 0xffffff8008e01120
0xffffff9227933c90 : 0xffffff8008e4f03c
0xffffff9227933dc0 : 0xffffff8008e4edbc
0xffffff9227933e20 : 0xffffff8008dff6f6
0xffffff9227933e30 : 0xffffff7f8c92a96d
0xffffff9227933eb0 : 0xffffff7f8c9387c5
0xffffff9227933ee0 : 0xffffff7f8c951432
0xffffff9227933fa0 : 0xffffff8008e004f7
      Kernel Extensions in backtrace:
         com.apple.filesystems.smbfs(3.
            dependency: com.apple.kec.c
            dependency: com.apple.kext.

BSD process name corresponding to curre

Mac OS version:
17D047

Kernel version:
Darwin Kernel Version 17.4.0: Sun Dec 1
Kernel UUID: 1BD901F1-4A03-3FF1-AE34-C2
Kernel slide:     0x0000000008a00000
Kernel text base: 0xffffff8008c00000
__HIB  text base: 0xffffff8008b00000
System model name: MacBookPro13,3 (Mac-

System uptime in nanoseconds: 100763845
last loaded kext at 100720886008863: co
    3 (addr 0xffffff7f91c7c000, size 32
last unloaded kext at 97218187654957: c
    5.0.0 (addr 0xffffff7f91c62000, siz
loaded kexts:
net.telestream.driver.TelestreamAudio 1
com.apple.filesystems.smbfs   3.2.1
com.apple.driver.AGPM110.23.30
com.apple.driver.ApplePlatformEnabler 2
com.apple.driver.X86PlatformShim  1.0.0
```

Hide Details

```
[  290.719853] Stack: c07ca1c0 00000000 c07ca1b8 c17ca240 c07ca1b8 c17ca1l
c180 c01496c9
[  290.720109]        00000001 c17ca240 53447380 0000003d 00000001 0000008
a240 52134680
[  290.720364]        0000003d fffff1b1 c014fe65 00000000 c049c120 525676a0
003d fffff1b3
[  290.720620] Call Trace:
[  290.720699]  [<c01496c9>] hrtimer_start+0xb9/0x140
[  290.720780]  [<c014fe65>] tick_nohz_stop_sched_tick+0x225/0x300
[  290.720868]  [<c010a930>] do_IRQ+0x40/0x70
[  290.720942]  [<c0108def>] common_interrupt+0x23/0x28
                             lt_idle+0x0/0x60
                             e_safe_halt+0x2/0x10
                             lt_idle+0x3c/0x60
                             dle+0x73/0xd0
                             _kernel+0x31f/0x3b0
                             m_bootoption+0x0/0x1f0
                             =====
                             00 00 00 57 31 ff 56 89 c6 53 83 ec 0c 89 54
                             9 14 24 8b 0b 85 c9 74 1d 8b 56 10 <3b> 51 10
                             6 8d 59 08 89 cf 8b
                             queue_hrtimer+0x29/0x100 SS:ESP 0068:c043bedc
                             fcdd353fb07 ]---
                             yncing: Attempted to kill the idle task!
```

```
                 __                    __   __    _______
                /  |____  ________ ___ /  | /  \  / ____/ /
               / ||_/ _ \/ ___/ __ \/ _ \/ / / /|\ / /_ / /
              / ,< /  __/ /  / / / /  __/ / /___\/ /__/ / /__
             /_/|_|\___/_/  /_/ /_/\___/_/ /_/   \_/_/  /____/

                    * x86-64 kernel AFL *        (  1 Processes)

Runtime:   000:00:00:13   | Performance: [||||||||||||||]  876 t/s
Last Path: 000:00:00:09   |
Bitmap:    01.2b/ 00.0%   |        Fuzzing Technique Progress
Blacklisted:     0/   0   | Bitflipping: [***************]   2.5K
                          | Arithmetic:  [***************]    36K
Cycles:               0   | Interesting: [***************]   4.9K
Level:            1/   1  | Havoc:       [***********    ]   4.1K
Favs:    2/   2  (100%)   | Splicing:    [               ]   4.1K
Pending:          1/   1  |
Skipped:          1/   1  | Panic:       0    (0)   CPU:     27.2%
Payload-Size:       112B  | KASan:       0    (0)   RAM:     05.1%
Total:              5.7K  | Timeout:     0    (0)         HAVOC
```

QEMU

Machine View

```
A problem has been detected and windows has been shut down to prevent damage
to your computer.

Modification of system code or a critical data structure was detected.

If this is the first time you've seen this Stop error s
```

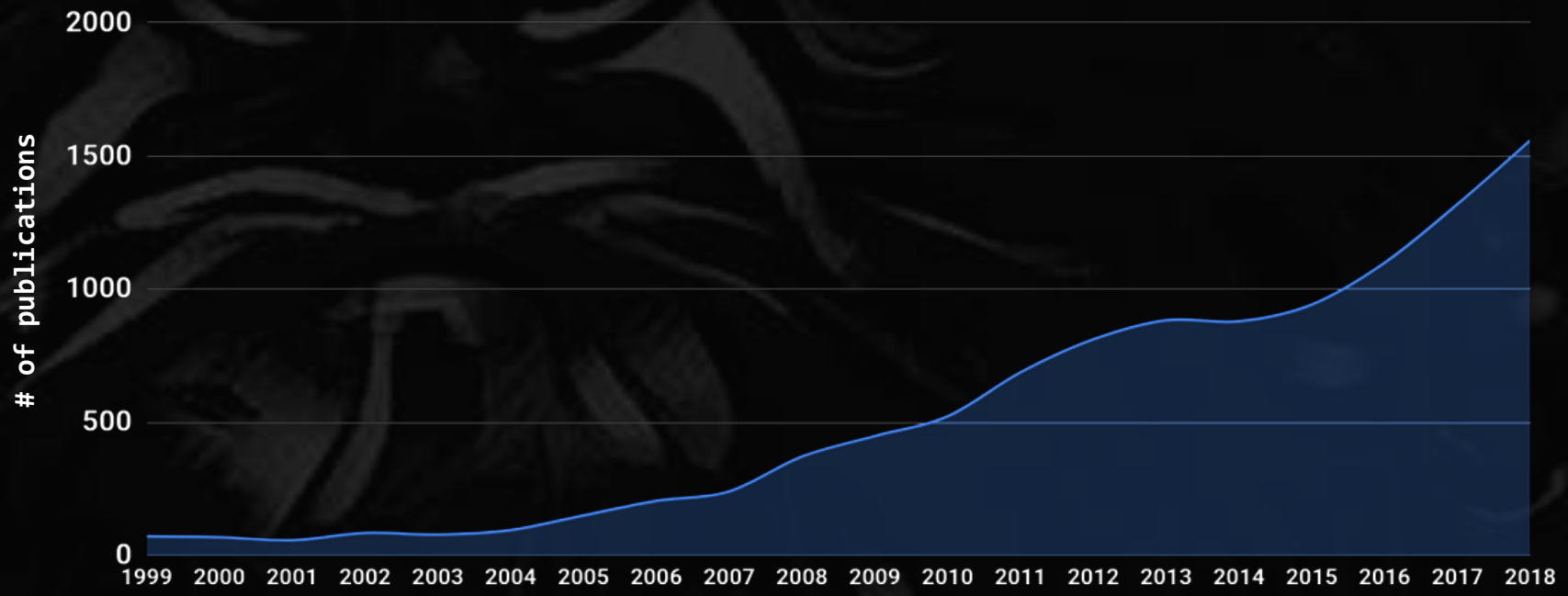
If this screen appears again, follow
these steps:
Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.
If problems continue, disable or remove any newly installed hardware
or software. Disable memory options such as caching or shadowing.
If your computer is very slow or hangs, restart your computer, press F8
to select Advanced Startup Options, and then
select Safe Mode.
Technical information:
*** STOP: 0x00000005 (0x00000000B,0x00000002,0x00000000,0x82F5F87E)
Collecting data for crash dump ...
Initializing disk for crash dump ...
Beginning dump of physical memory.
Dumping physical memory to disk: 100
Physical memory dump complete.
Contact your system admin or technical support group for further assistance.

Most Popular Languages in July 2019

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

https://www.tiobe.com/tiobe-index/

Fuzzing is Very Hot Today!



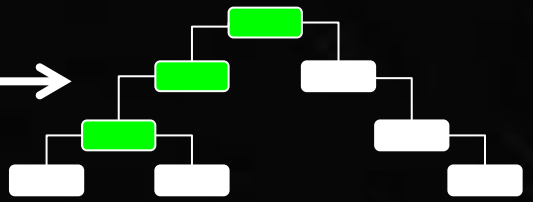
OSS-Fuzz Project

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

Open Issues Count per Month

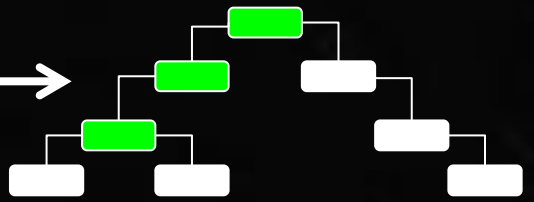
Downsides. Volatile Paths

AAAAAAAAA

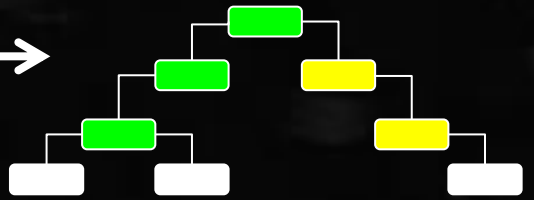


Downsides. Volatile Paths

AAAAAAAAA

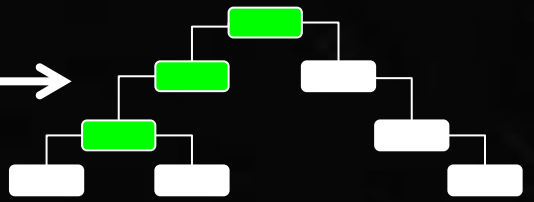


ABAAAAAAA

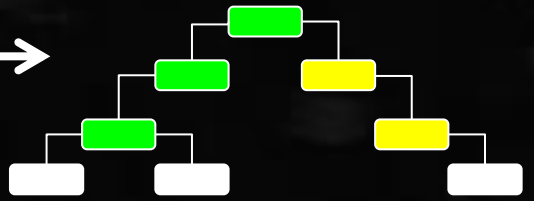


Downsides. Volatile Paths

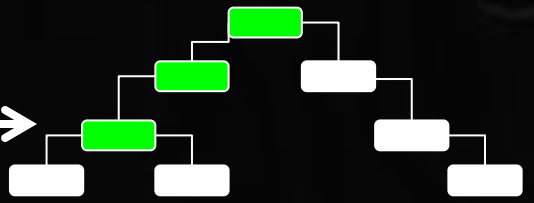
AAAAAAAAA



ABAAAAAAA

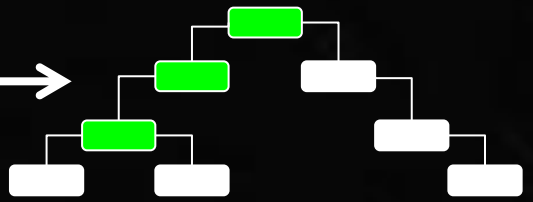


ABAAAAAAA

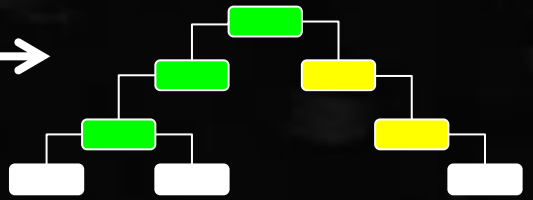


Downsides. Volatile Paths

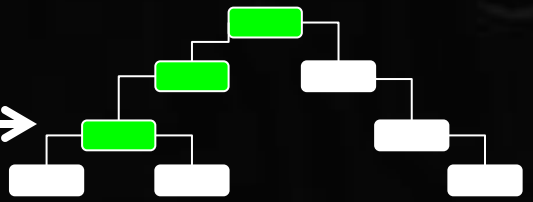
AAAAAAAAA



ABAAAAAAA

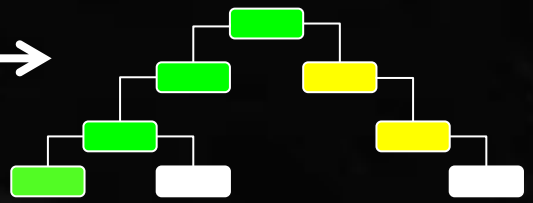


ABAAAAAAA

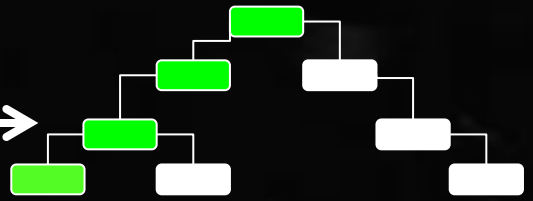


Downsides. Volatile Paths

ABAAAAAAA



ABAAAAAAA



Downsides. Volatile Paths

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

┌─ process timing ─────────────────────────────────────┬─ overall results ─────┐
│        run time : 0 days, 0 hrs, 50 min, 5 sec       │  cycles done : 0      │
│   last new path : 0 days, 0 hrs, 3 min, 40 sec       │  total paths : 203    │
│ last uniq crash : none seen yet                      │ uniq crashes : 0      │
│  last uniq hang : none seen yet                      │   uniq hangs : 0      │
├─ cycle progress ────────────────────┬─ map coverage ─┴───────────────────────┤
│  now processing : 18 (8.87%)        │    map density : 7.19% / 10.44%        │
│ paths timed out : 0 (0.00%)         │ count coverage : 1.56 bits/tuple       │
├─ stage progress ────────────────────┼─ findings in depth ────────────────────┤
│  now trying : interest 32/8         │ favored paths : 106 (52.22%)           │
│ stage execs : 1008/2892 (34.85%)    │  new edges on : 132 (65.02%)           │
│ total execs : 136k                  │ total crashes : 0 (0 unique)           │
│  exec speed : 44.90/sec (slow!)     │  total tmouts : 0 (0 unique)           │
├─ fuzzing strategy yields ───────────┴───────────────┬─ path geometry ────────┤
│   bit flips : 13/4592, 2/4580, 2/4556               │    levels : 2          │
│  byte flips : 0/574, 0/562, 0/538                   │   pending : 192        │
│ arithmetics : 13/32.0k, 0/17.5k, 0/8583             │  pend fav : 98         │
│  known ints : 2/2596, 3/12.2k, 8/18.2k              │ own finds : 137        │
│  dictionary : 0/0, 0/0, 0/4674                      │  imported : n/a        │
│       havoc : 94/22.6k, 0/0                         │ stability : 44.90%     │
│        trim : 12.23%/177, 0.00%                     ├────────────────────────┘
└─────────────────────────────────────────────────────┘          [cpu000:  2%]
^C
```

Downsides. Parallelization algorithm

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

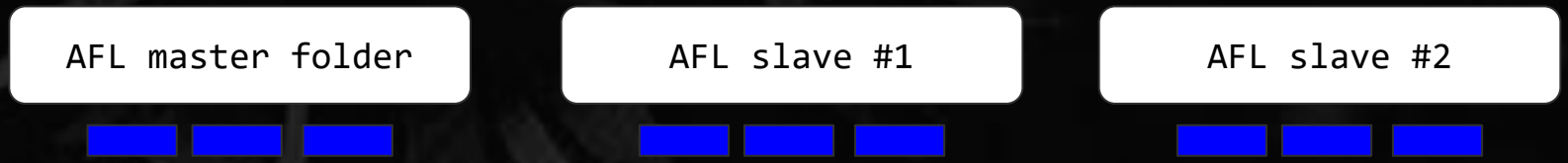
AFL master folder

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

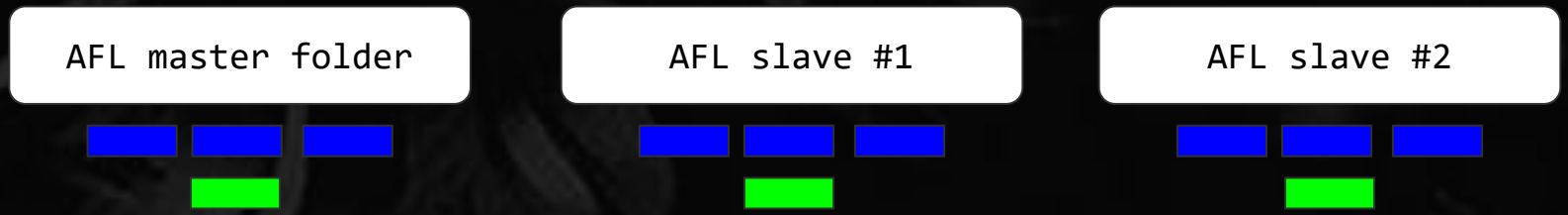


Downsides. Parallelization algorithm

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

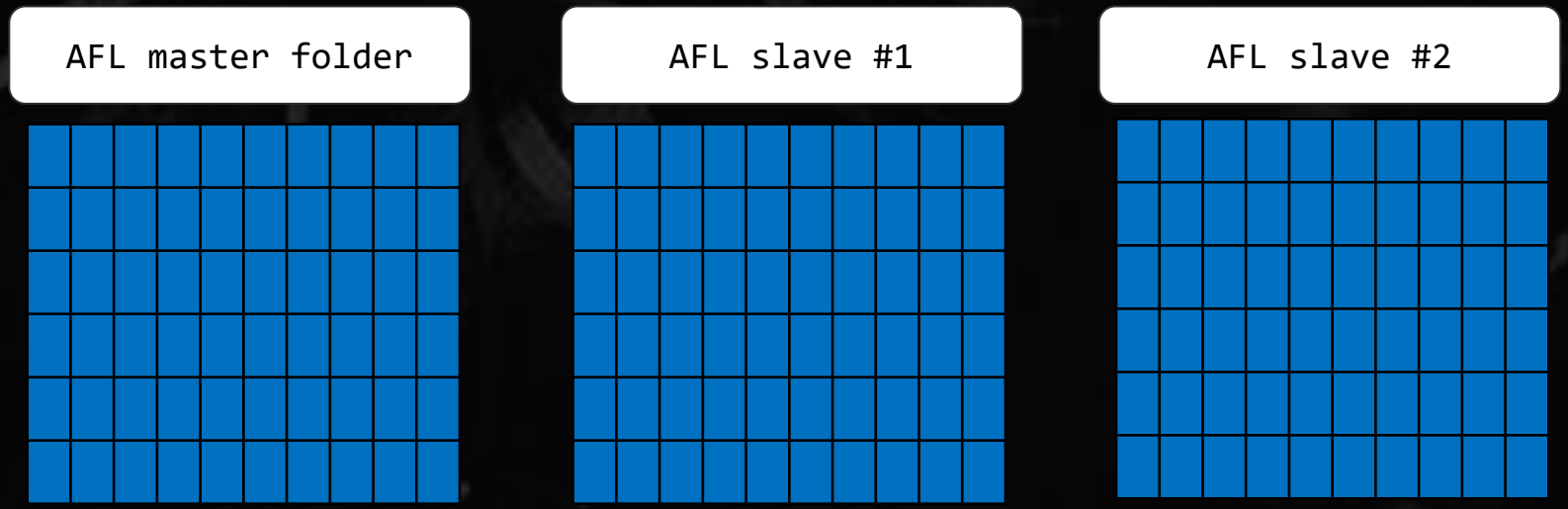
Downsides. Parallelization algorithm

- 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

```
                WinAFL 1.11 based on AFL 2.43b

process timing ------------------------------------+- overall results ----+
       run time : 0 days, 1 hrs, 19 min, 10 sec      |  cycles done : 0      |
  last new path : 0 days, 0 hrs, 0 min, 52 sec       |  total paths : 1275   |
last uniq crash : none seen yet                      | uniq crashes : 0      |
last uniq hang : 0 days, 1 hrs, 14 min, 18 sec       |   uniq hangs : 1      |
cycle progress ---------------------+- map coverage -+-----------------------+
now processing : 132 (10.35%)       |    map density : 2.22% / 11.64%       |
paths timed out : 0 (0.00%)         | count coverage : 2.67 bits/tuple      |
stage progress ---------------------+ findings in depth ---------------------+
now trying : arith 8\8              | favored paths : 222 (17.41%)          |
stage execs : 40.0k/61.7k (64.90%)  |  new edges on : 305 (23.92%)          |
total execs : 2.68M                 | total crashes : 0 (0 unique)          |
exec speed : 924.1/sec              |  total tmouts : 61 (1 unique)         |
fuzzing strategy yields -----------+---------------+- path geometry -------+
  bit flips : 660/115k, 80/115k, 68/115k             |    levels : 3         |
 byte flips : 7/14.4k, 17/14.4k, 17/14.4k            |   pending : 1261      |
arithmetics : 187/745k, 0/8369, 0/0                  |  pend fav : 215       |
 known ints : 32/80.5k, 27/453k, 21/533k             | own finds : 1274      |
 dictionary : 0/0, 0/0, 37/410k                      |  imported : n/a       |
      havoc : 118/5125, 0/0                          | stability : 73.55%    |
       trim : 6.61%/7080, 0.00%                      +-----------------------+
-----------------------------------------------------+
```

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:
  - *Valentin J.M. Manes, Hyung Seok Han, Choongwoo Han, Sang Kil Cha, Manuel Egele, Edward J. Schwartz, Maverick Woo Fuzzing: Art, Science, and Engineering. arXiv:1812.00140 preprint.*
- Some Presentations at DEF CON/BlackHat:
  - *Mateusz Jurczyk. Effective File Format Fuzzing - Thoughts, Techniques and Results. BlackHat EU London. 2016.*
  - *Kang Li. AFL's Blindspot and How to Resist AFL Fuzzing for Arbitrary ELF Binaries. BlackHat USA 2018.*
  - *Jonathan Metzman. Going Beyond Coverage-Guided Fuzzing with Structured Fuzzing. Black Hat USA 2019.*

State-of-the-art Userland Fuzzers

	AFL winAFL	HongFuzz	libFuzzer	Desired fuzzer
Network fuzzing	No (Unix) Yes (Windows)	Yes	No	Yes (all platforms)
Volatile Paths	No	No	No	Yes
Multiple Mutation Strategies	No	No	No	Yes
Share over network	Partial	No	No	Yes
Supported Platform	Linux Windows	Open/NetBSD GNU/Linux Windows/Cygwin Android OS X	Anywhere where LLVM exist	Anywhere where Python exist
Language	C	C	C	Python

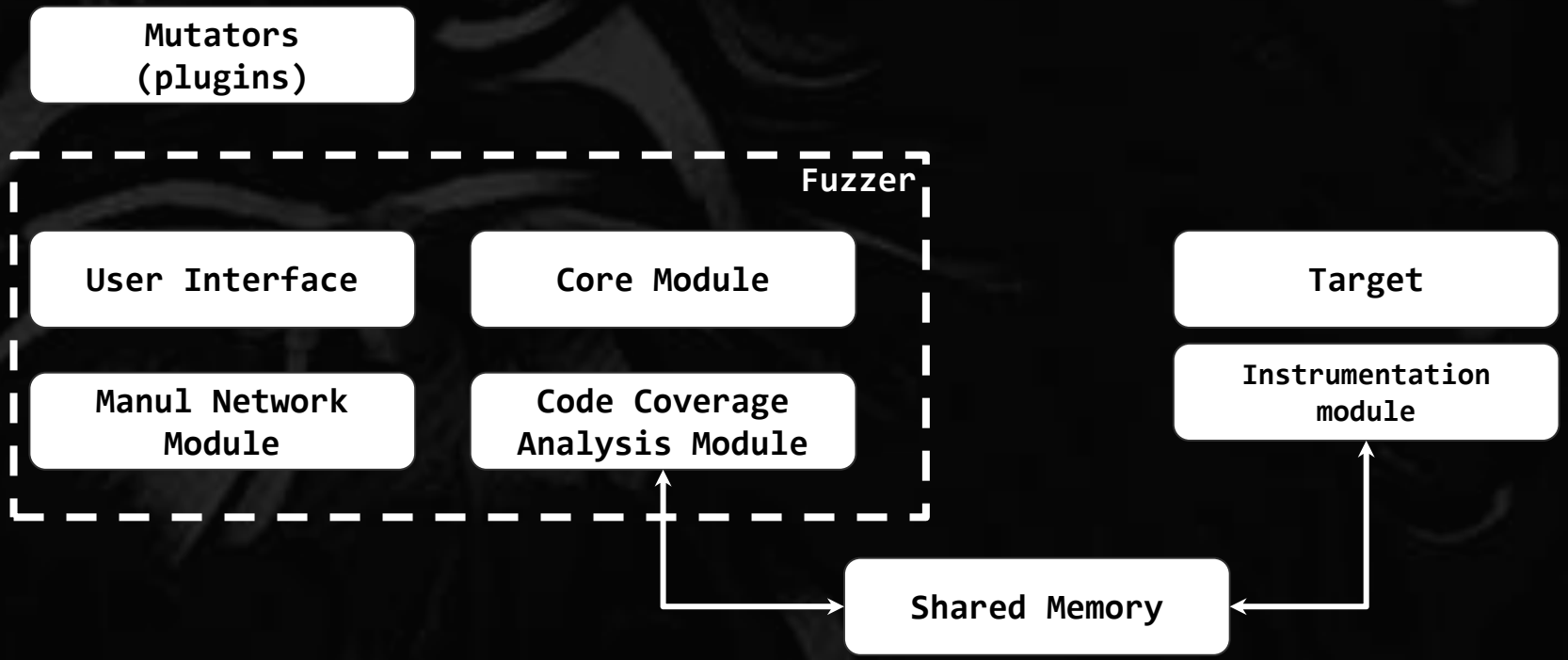
Manul Overview

- 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?

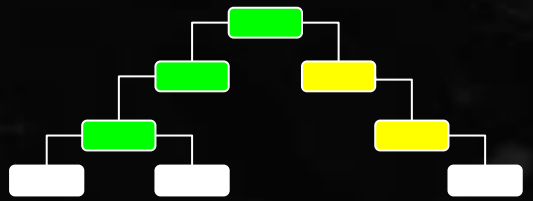
Pallas’s Cat (lat. Otocolobus Manul)

Manul Architecture



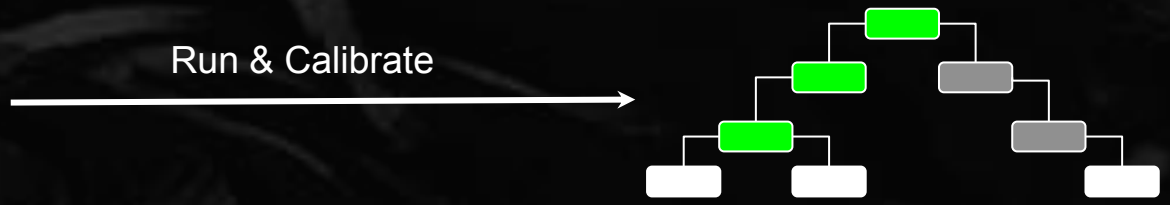
Volatile Paths Detection

ABAAAAAAA → Run & Calibrate

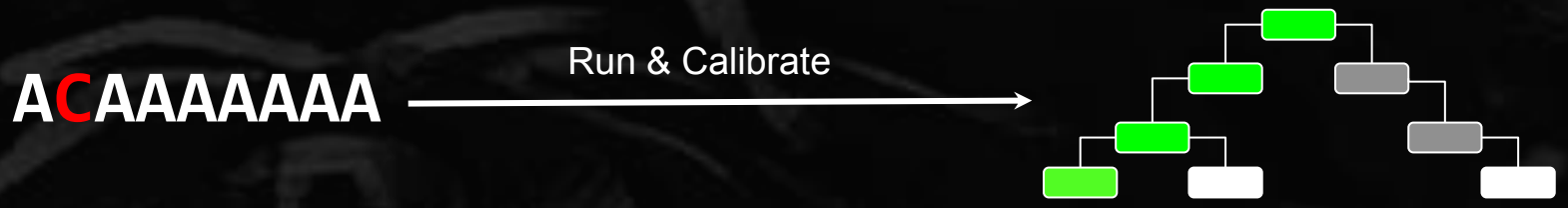


Volatile Paths Detection

ABAAAAAAA



Volatile Paths Detection



https://www.usenix.org/system/files/conference/usenixsecurity17/sec17-schumilo.pdf

Parallel fuzzing. Python Multiprocessing

Main Process

Corpus:

Parallel fuzzing. Python Multiprocessing

Instance #1

Instance #2

Instance #3

Main Process

Corpus:

Parallel fuzzing. Python Multiprocessing

Instance #1

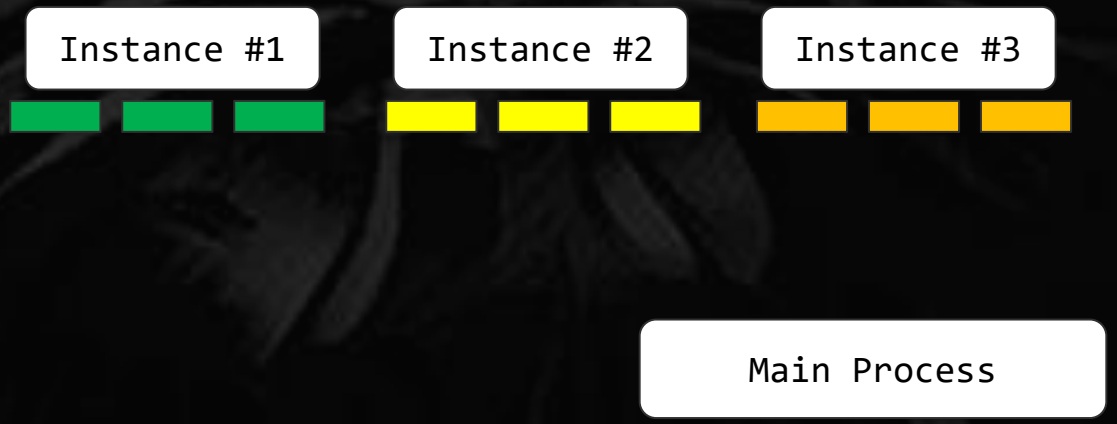
Instance #2

Instance #3

Main Process

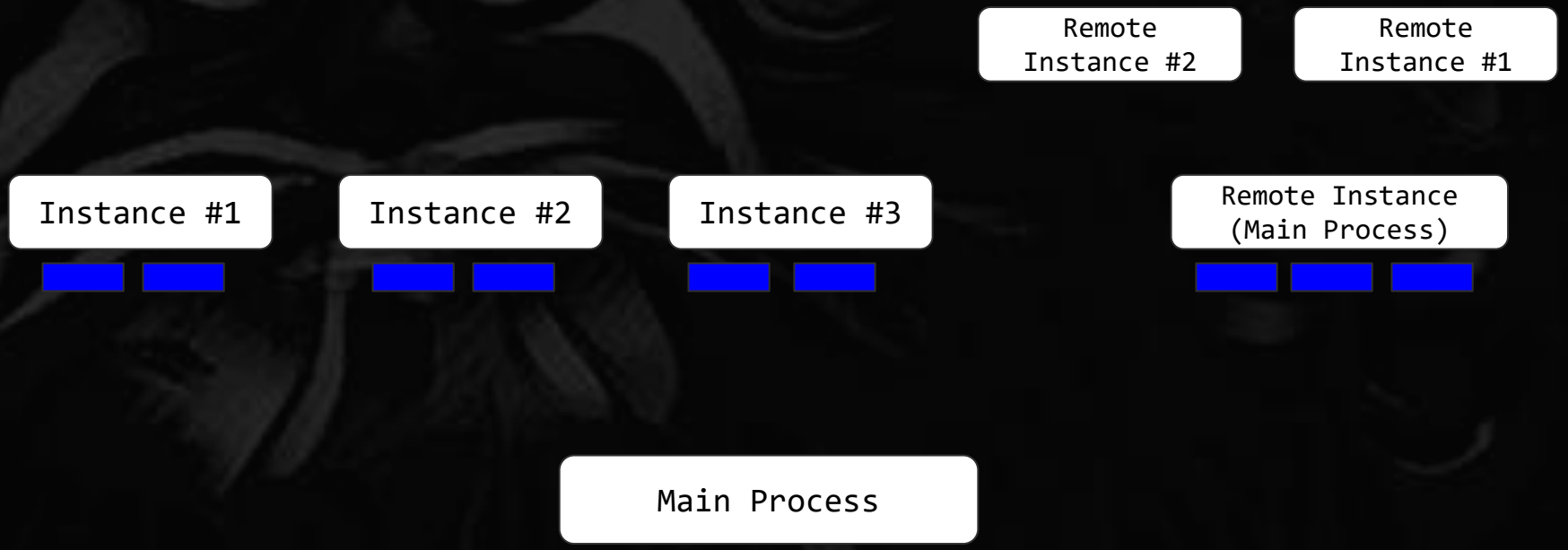
Corpus:

Parallel fuzzing. Python Multiprocessing

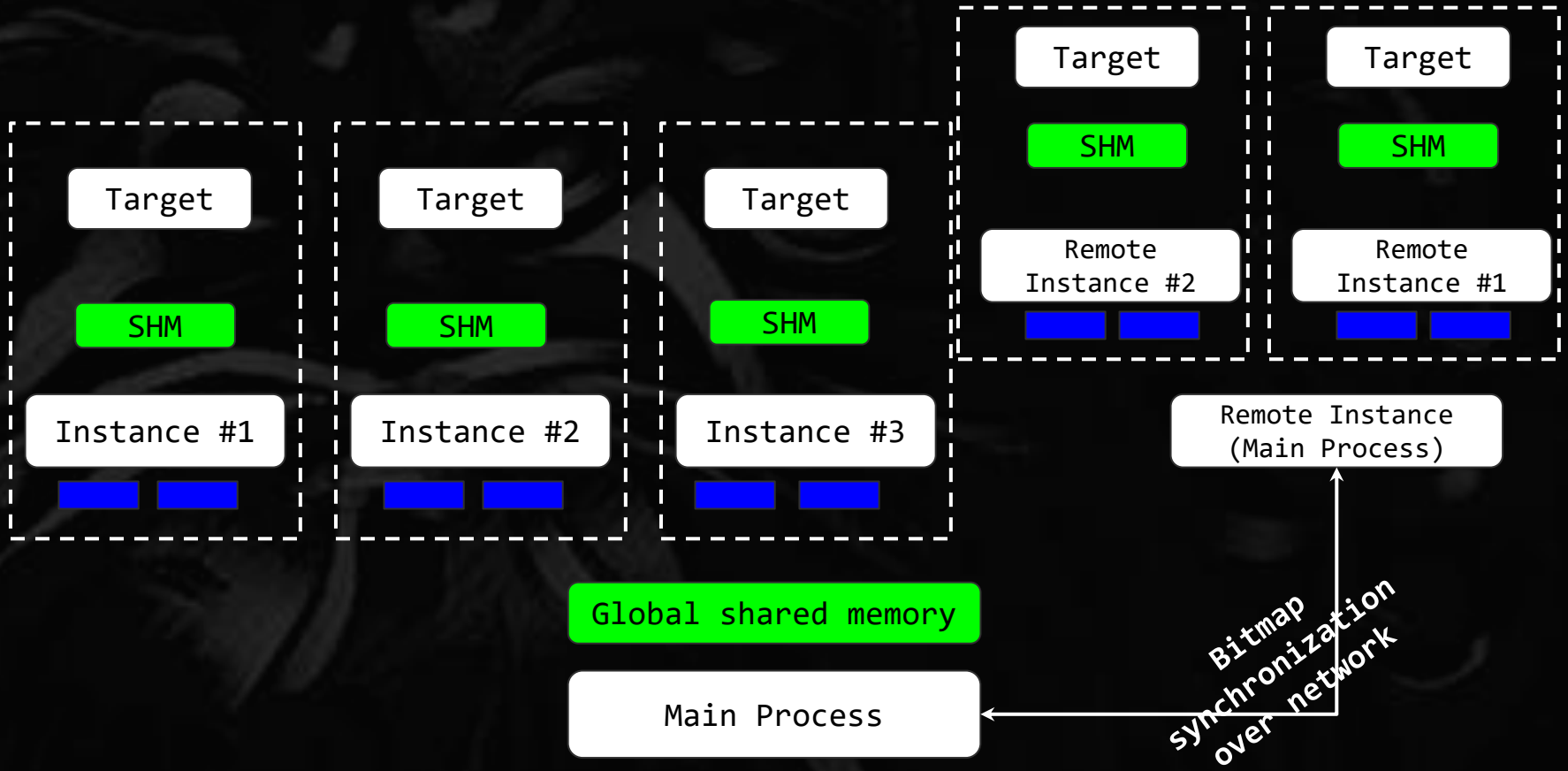


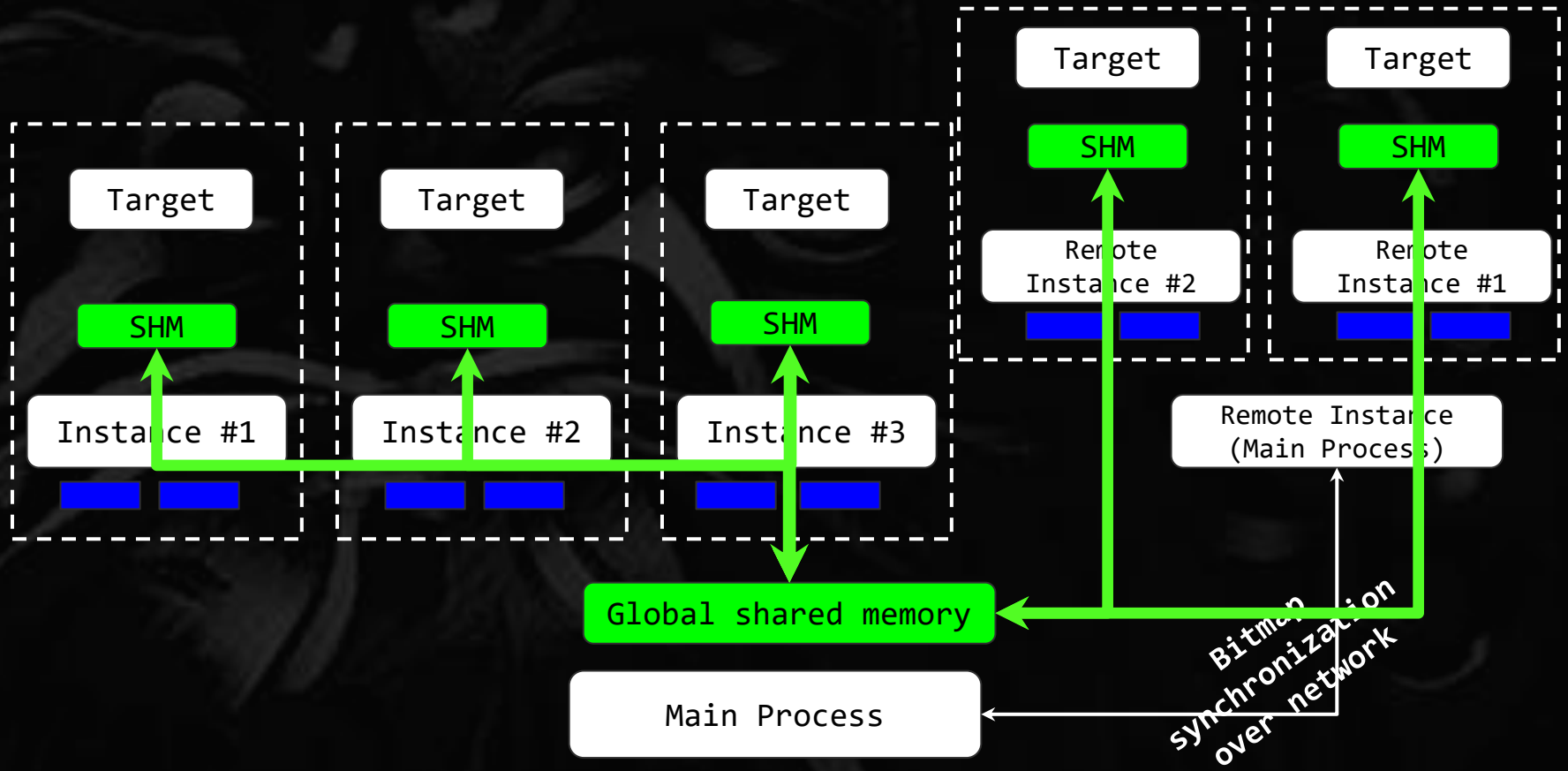
Corpus:

Parallel fuzzing. Python Multiprocessing



Parallel fuzzing. Python Multiprocessing


Target
SHM
Instance #1
Target
SHM
Instance #2
Target
SHM
Instance #3
Target
SHM
Remote Instance #2
Target
SHM
Remote Instance #1
Remote Instance (Main Process)
Global shared memory
Main Process
Bitmap synchronization over network


Target
SHM
Instance #1
Target
SHM
Instance #2
Target
SHM
Instance #3
Target
SHM
Remote Instance #2
Target
SHM
Remote Instance #1
Remote Instance (Main Process)
Global shared memory
Main Process
Bitmap synchronization over network

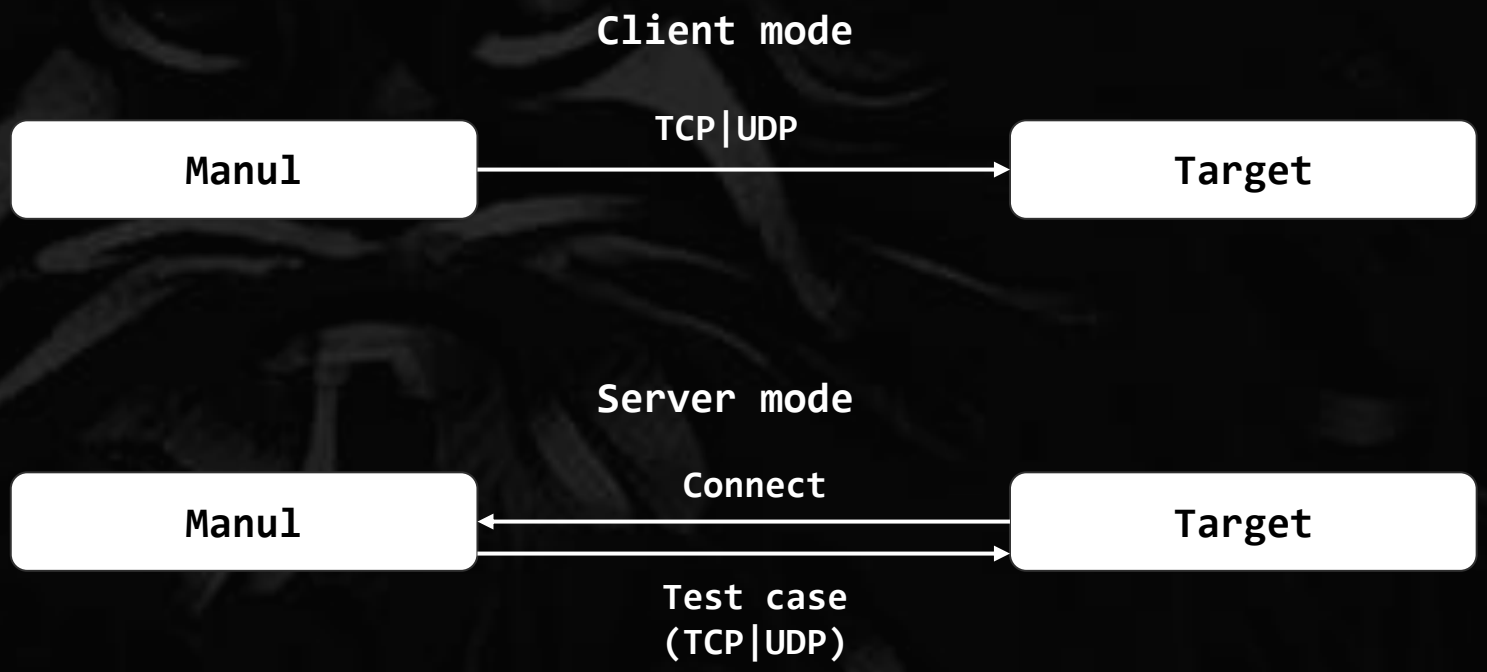
Third Party Mutators

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

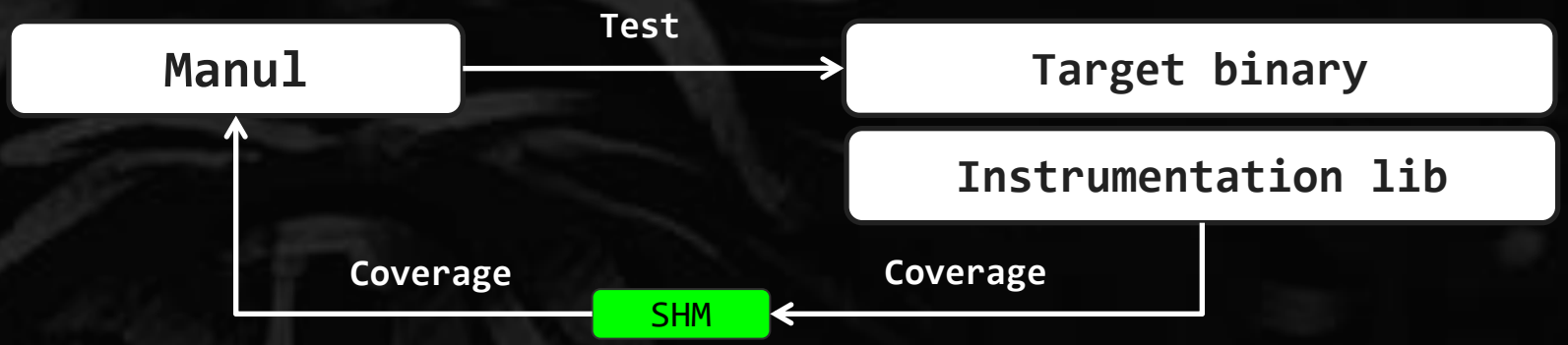
Custom Python Mutator:

- `def init(fuzzer_id)`
- `def mutate(data_to_mutate)`

Network Application Fuzzing (Experimental)



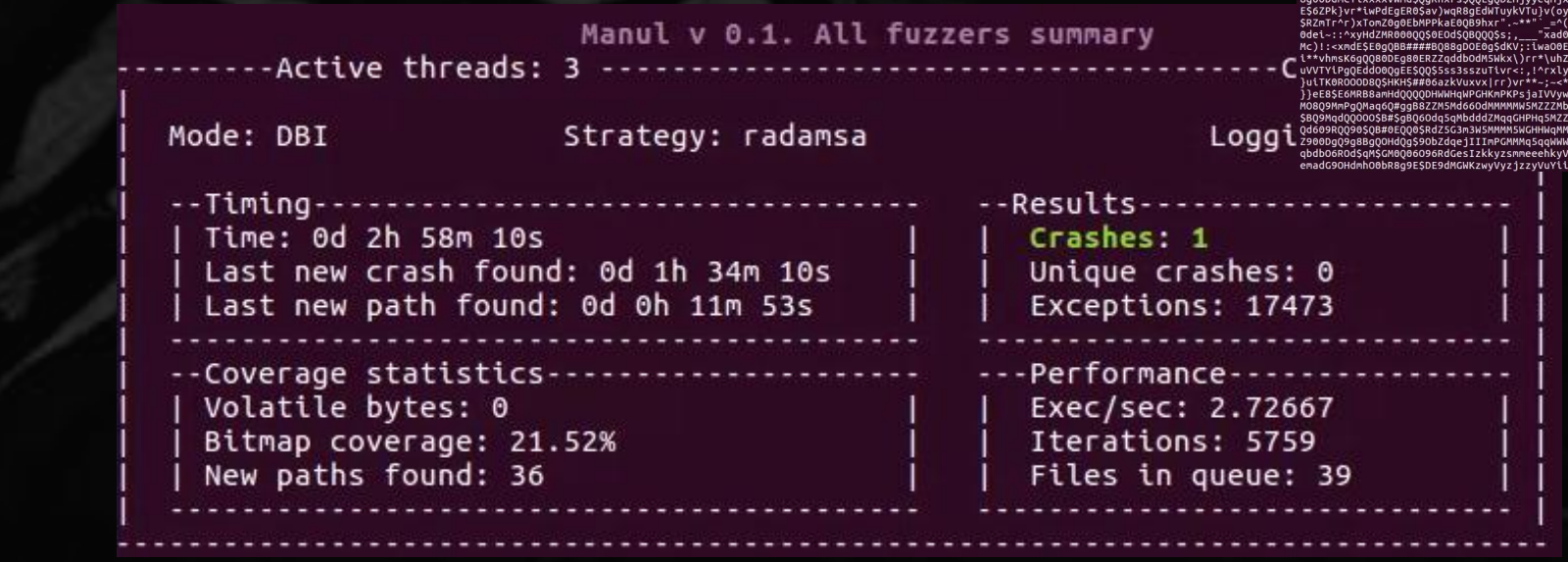
Blackbox Binaries Fuzzing



Windows: DynamoRIO: ~x30 overhead
Linux: Intel Pin: ~x45 overhead
DynamoRIO: ~x20 overhead

python.exe	10.69	7 916 K	16 364 K	16932	Python	Python Software Foundation
python.exe	0.13	7 652 K	16 028 K	8152	Python	Python Software Foundation
cmd.exe		3 376 K	3 476 K	15476	Обработчик команд Windo...	Microsoft Corporation
drmun.exe		1 056 K	2 984 K	248	DynamoRIO configure-and-r...	DynamoRIO developers
test64.exe	12.81	2 044 K	5 168 K	16724		
cmd.exe	0.45	3 380 K	3 436 K	6048	Обработчик команд Windo...	Microsoft Corporation
drmun.exe	0.46	1 084 K	3 020 K	20844	DynamoRIO configure-and-r...	DynamoRIO developers
test64.exe	9.48	1 720 K	3 880 K	11492		
cmd.exe	0.43	3 344 K	3 424 K	15908	Обработчик команд Windo...	Microsoft Corporation
drmun.exe	0.41	1 092 K	3 028 K	19076	DynamoRIO configure-and-r...	DynamoRIO developers
test64.exe	0.79	1 044 K	2 980 K	11548		
python.exe		7 692 K	16 072 K	12268	Python	Python Software Foundation
cmd.exe	0.38	3 384 K	3 436 K	19920	Обработчик команд Windo...	Microsoft Corporation
drmun.exe	0.33	1 088 K	3 028 K	11052	DynamoRIO configure-and-r...	DynamoRIO developers
test64.exe	8.66	1 732 K	3 892 K	15716		
cmd.exe		3 376 K	3 476 K	18968	Обработчик команд Windo...	Microsoft Corporation
drmun.exe		1 084 K	3 024 K	17976	DynamoRIO configure-and-r...	DynamoRIO developers
test64.exe	17.51	2 076 K	5 376 K	11344		
python.exe	0.14	7 716 K	16 088 K	21136	Python	Python Software Foundation
cmd.exe		3 344 K	3 464 K	5284	Обработчик команд Windows	icrosoft Corporation
drmun.exe		1 008 K	2 952 K	10676	DynamoRIO configure-and-r...	DynamoRIO developers
test64.exe	12.87	2 076 K	5 376 K	3720		
cmd.exe	0.51	3 344 K	3 424 K	3404	Обработчик команд Windo...	Microsoft Corporation
drmun.exe	0.44	1 092 K	3 032 K	2940	DynamoRIO configure-and-r...	DynamoRIO developers
test64.exe	11.27	1 752 K	3 908 K	10164		
cmd.exe	0.49	3 376 K	3 436 K	11096	Обработчик команд Windo...	Microsoft Corporation
drmun.exe	0.41	1 052 K	2 988 K	13236	DynamoRIO configure-and-r...	DynamoRIO developers
test64.exe	1.92	1 036 K	2 980 K	10964		
python.exe	0.16	7 608 K	15 984 K	20600	Python	Python Software Foundation
cmd.exe		3 372 K	3 472 K	20676	Обработчик команд Windo...	Microsoft Corporation
drmun.exe		1 092 K	3 032 K	5912	DynamoRIO configure-and-r...	DynamoRIO developers
test64.exe	12.82	1 720 K	4 036 K	18708		

Interface & Logo



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)

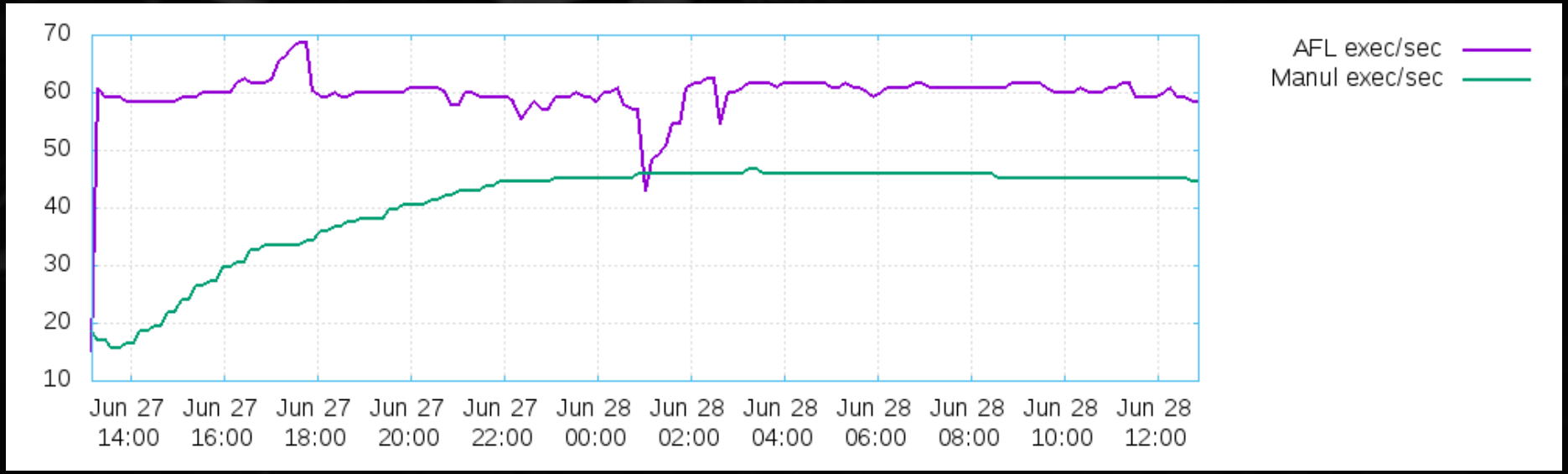
Case Study I. Poppler

- 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)

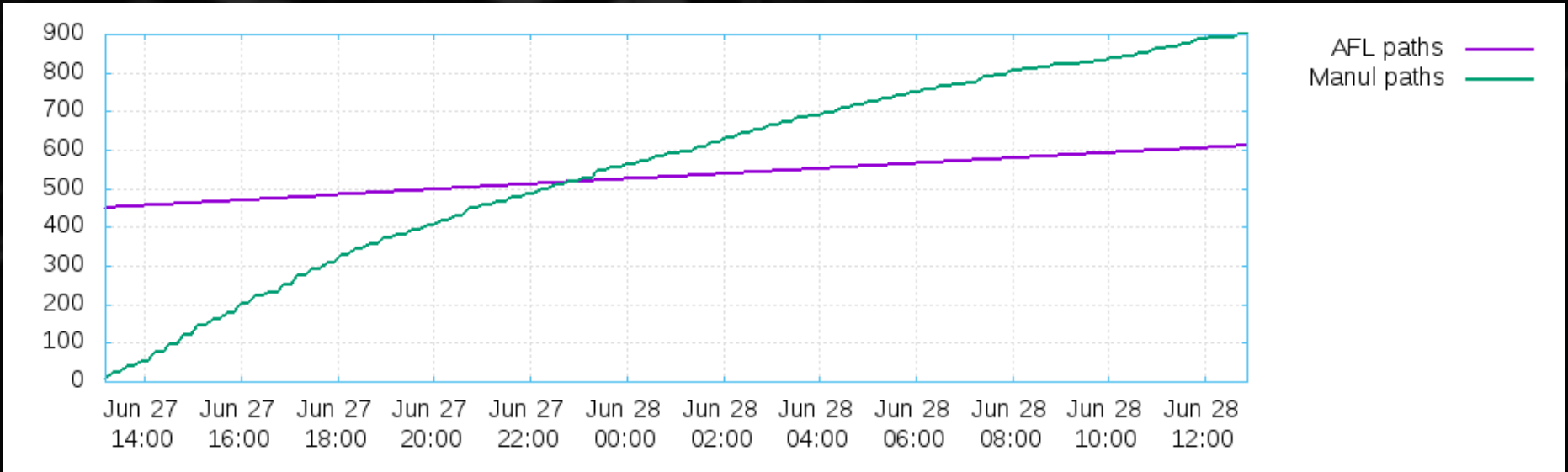
Case Study I. Poppler. Fuzzing Setup

- 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

Case Study I. Poppler. CVE 2019-9631

```
1  static void downsample_row_box_filter (int start, int width, uint32_t *src, uint32_t *dest, int coverage[], int pixel_coverage)
2  {
3      <---truncated----->
4      while (x < start + width)
5      {
6          int box = 1 << FIXED_SHIFT;
7          int start_coverage = coverage[x];
8
9          a = ((*src >> 24) & 0xff) * start_coverage;
10         r = ((*src >> 16) & 0xff) * start_coverage;
11         g = ((*src >>  8) & 0xff) * start_coverage;
12         b = ((*src >>  0) & 0xff) * start_coverage;
13         src++;
14         x++;
15         box -= start_coverage;
16
17         while (box >= pixel_coverage)
18         {
19             a += ((*src >> 24) & 0xff) * pixel_coverage; // <---- overrun happens here
20             r += ((*src >> 16) & 0xff) * pixel_coverage;
21             g += ((*src >>  8) & 0xff) * pixel_coverage;
22             b += ((*src >>  0) & 0xff) * pixel_coverage;
23             src++;
24
25             box -= pixel_coverage;
26         }
27         if (box > 0)
28         {
29             a += ((*src >> 24) & 0xff) * box;
30             r += ((*src >> 16) & 0xff) * box;
31             g += ((*src >>  8) & 0xff) * box;
32             b += ((*src >>  0) & 0xff) * box;
33         }
34
35         a >>= FIXED_SHIFT;
36         r >>= FIXED_SHIFT;
37         g >>= FIXED_SHIFT;
38         b >>= FIXED_SHIFT;
39
40         *dest = (a << 24) | (r << 16) | (g << 8) | b;
41         dest++;
42     }
43 }
```

Case Study I. Poppler. CVE 2019-9631

```
1   static void downsample_row_box_filter (int start, int width, uint32_t *src, uint32_t *dest, int coverage[], int pixel_coverage)
2   {
3       <---truncated----->
4       while (x < start + width)
5       {
6           int box = 1 << FIXED_SHIFT;
7           int start_coverage = coverage[x];
8
9           a = ((*src >> 24) & 0xff) * start_coverage;
10          r = ((*src >> 16) & 0xff) * start_coverage;
11          g = ((*src >>  8) & 0xff) * start_coverage;
12          b = ((*src >>  0) & 0xff) * start_coverage;
13          src++;
14          x++;
15          box -= start_coverage;
16
17          while (box >= pixel_coverage)
18          {
19              a += ((*src >> 24) & 0xff) * pixel_coverage; // <---- overrun happens here
20              r += ((*src >> 16) & 0xff) * pixel_coverage;
21              g += ((*src >>  8) & 0xff) * pixel_coverage;
22              b += ((*src >>  0) & 0xff) * pixel_coverage;
23              src++;
24
25              box -= pixel_coverage;
26          }
27          if (box > 0)
28          {
29              a += ((*src >> 24) & 0xff) * box;
30              r += ((*src >> 16) & 0xff) * box;
31              g += ((*src >>  8) & 0xff) * box;
32              b += ((*src >>  0) & 0xff) * box;
33          }
34
35          a >>= FIXED_SHIFT;
36          r >>= FIXED_SHIFT;
37          g >>= FIXED_SHIFT;
38          b >>= FIXED_SHIFT;
39
40          *dest = (a << 24) | (r << 16) | (g << 8) | b;
41          dest++;
42      }
43  }
```

Case Study I. Poppler. CVE 2019-9959

```
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);
```

Case Study I. Poppler. CVE 2019-9959

```
inline unsigned char *toUnsignedChars(int *length, int initialSize = 4096, int sizeIncrement = 4096)
{
  int readChars;
  unsigned char *buf = (unsigned char *)gmalloc(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 *)grealloc(buf, size);
      } else {
        continueReading = false;
      }
    } else {
      continueReading = false;
    }
  }
  return buf;
}
```

Case Study I. Poppler. CVE 2019-7310

```
XRefEntry *XRef::getEntry(int i, bool complainIfMissing)
{
  if (i >= size || entries[i].type == xrefEntryNone) {

    if ((!xRefStream) && mainXRefEntriesOffset) {
      if (unlikely(i >= capacity)) {
        error(errInternal, -1, "Request for out-of-bounds XRef entry [{0:d}]", i);
        return &dummyXRefEntry;
      }

      if (!parseEntry(mainXRefEntriesOffset + 20*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 powerfull 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. (X.X High). In Zeek IDS, there is a DoS in Kerberos protocol parser in analyzer/protocol/krb/KRB.cc

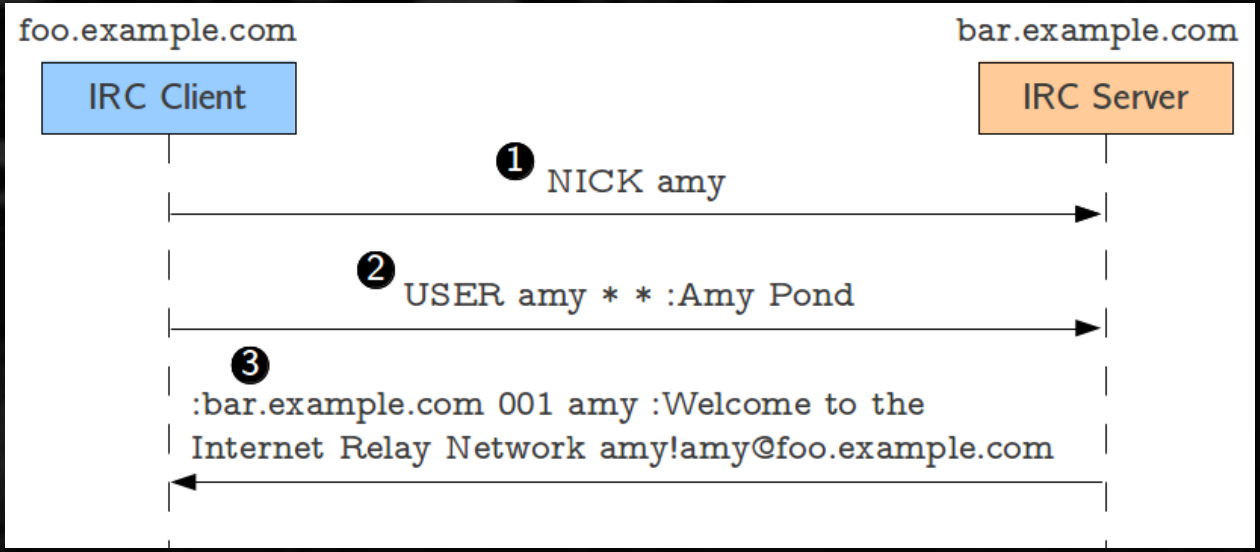
CVE-2018-16807

```
#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)
```

```diff
-       c$krb$service      = msg$service_name;

        if ( msg?$from )
                c$krb$from = msg$from;
@@ -183,7 +184,8 @@ event krb_tgs_request(c: connection, msg: KDC_Request) &priority=5
                return;

        c$krb$request_type = "TGS";
-       c$krb$service = msg$service_name;

        if ( msg?$from )
                c$krb$from = msg$from;
        c$krb$till = msg$till;
```

```diff
+       if ( msg?$service_name )
+               c$krb$service      = msg$service_name;

        if ( msg?$from )
                c$krb$from = msg$from;
                return;

        c$krb$request_type = "TGS";
+       if ( msg?$service_name )
+               c$krb$service = msg$service_name;
        if ( msg?$from )
                c$krb$from = msg$from;
        c$krb$till = msg$till;
```

IRC Protocol



CVE 2018-16807. Packet Example

```
255  -                    // Remove nick name.
256  -                    parts.erase(parts.begin());
257  -                    if ( parts.size() < 2 )
```

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

Bugs	Project
CVE-2019-6931, CVE-2019-7310, CVE-2019-9959	Poppler for Linux
CVE-2018-17019, CVE-2018-16807, CVE-2019-12175	Zeek for Linux
CVE-2019-XXXX, CVE-2019-XXXX Awaiting assignment from MITRE and fix from maintainer	7-Zip 19.00 for Windows
CVE-2019-XXXX, CVE-2019-XXXX, CVE-2019-XXXX Awaiting assignment from MITRE and fix from maintainer	p7zip 16.02 for Linux
CVE-2019-XXXX, CVE-2019-XXXX Awaiting assignment from MITRE and fix from maintainer	Unarchiver for MacOS

Discussion & Future Work

- AFL’s forkserver 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
  - pip install psutil & git clone https://github.com/mxmssh/manul

Thank you!

https://github.com/mxmssh/manul

Twitter: https://twitter.com/MShudrak

Linkedin: https://www.linkedin.com/in/mshudrak/