

# Building Advanced Coverage-guided Fuzzer for Program Binaries

NGUYEN Anh Quynh <[aquynh@gmail.com](mailto:aquynh@gmail.com)>  
WEI Lei

17/11/2017

Zero Nights, Moscow 2017

# Self-introduction

- NGUYEN Anh Quynh, PhD <aquynh @ gmail.com>
  - Nanyang Technological University
  - Operating System, Virtual Machine, Binary Analysis, etc
  - Reverse trilogy: Capstone, Unicorn & Keystone
- WEI Lei, PhD
  - Nanyang Technological University
  - ~ 60 CVEs in Adobe, Apple, PHP etc
  - ~ 50 bug bounties from iDefense VCP, TippingPoint ZDI, and HackerOne.

# Agenda

- Coverage-guided fuzzer
  - Background
  - Issues of public guided fuzzers
- Darko fuzzer
  - Features
  - Design & Implementation
- Demo & bugs found
- Conclusions

# Coverage-guided Fuzzer

# Fuzzer

- Automated software testing technique to find bugs
  - Feed craft input data to the program under test
  - Monitor for errors like crash/hang/memory leaking
  - Focus more on exploitable errors like memory corruption, info leaking
- Maximize code coverage to find bugs
- Blackbox fuzzing
- Whitebox fuzzing
- **Graybox fuzzing**

# Coverage-guided fuzzer

- Instrument target binary to collect coverage info
- Mutate the input to maximize the coverage
- Repeat above steps to find bugs
  - Proved to be very effective
    - Easier to use/setup & found a lot of bugs
  - Trending in fuzzing technology
    - American Fuzzy Lop (AFL) really changed the game

# Public guided fuzzers

- AFL
  - Requires source code for instrumentation build
  - Supports \*nix binary via emulation mode (Qemu)
- AFL-Cygwin
  - AFL ported to Windows via Cygwin
  - Slow, buggy & development stalled
- WinAFL
  - Windows fork - needs persistent mode support
- AFL-Dyninst
  - Static-based instrumentation struggle on complicated binaries
  - No Windows support

# Problems of public guided fuzzers

- Poor support for fuzzing binary
  - AFL emulation mode based on QEMU is limited
    - Only support Linux
    - Limitation of QEMU user mode emulation
  - Only WinAFL handles Windows closed source binaries
- Tricky to use
  - WinAFL persistent mode is really painful
- Suffer on performance & stability
  - DynamoRio is slow & fails to work on some large binaries
  - Needs persistent mode to perform well



# DARKO Fuzzer

# Darko design

- Motivation: no coverage-guided fuzzer for Windows (Dec 2015)
- Fork AFL fuzzing code & ported to Windows (Apr 2016)
  - Rewrite to work with our target instrumentation
- Support closed source binary for all platforms & architectures
  - To have a cross-platform/architecture fuzzer
  - Build our own instrumentation from scratch (Apr 2016)
    - Replaced with SKORPIO - multi-arch / platform (2017)
  - Support selective binary fuzzing
  - Support persistent mode
- Various other enhancements to AFL (2017)

# Darko features

- Pure software-based
- Cross-platform/architecture
  - Native compiled (MSVC on Windows, GCC/Clang on \*nix)
- Binary support
  - Full & selective binary fuzzing + Persistent mode
- Fast + stable
  - Stable & support all kind of binaries
  - Order of magnitude faster than DBI/Emulation approaches

# Darko implementation - Overview

- AFL-compatible instrumentation
- PoC: AFL-Cygwin + PIN Probe mode (Apr 2016)
  - Applicable to user-space 32-bit Windows binaries
  - Flexible test case post-processor
  - Found bugs in Adobe Reader, Windows Journal, etc
- Static analysis + dynamic binary rewriting (SMC)
  - Speed much better than full binary DBI
- Near native execution speed, ASLR / threading compatible
- Support Windows, Linux & MacOS
- Support for non-X86 architectures underway

# Challenges in static analysis

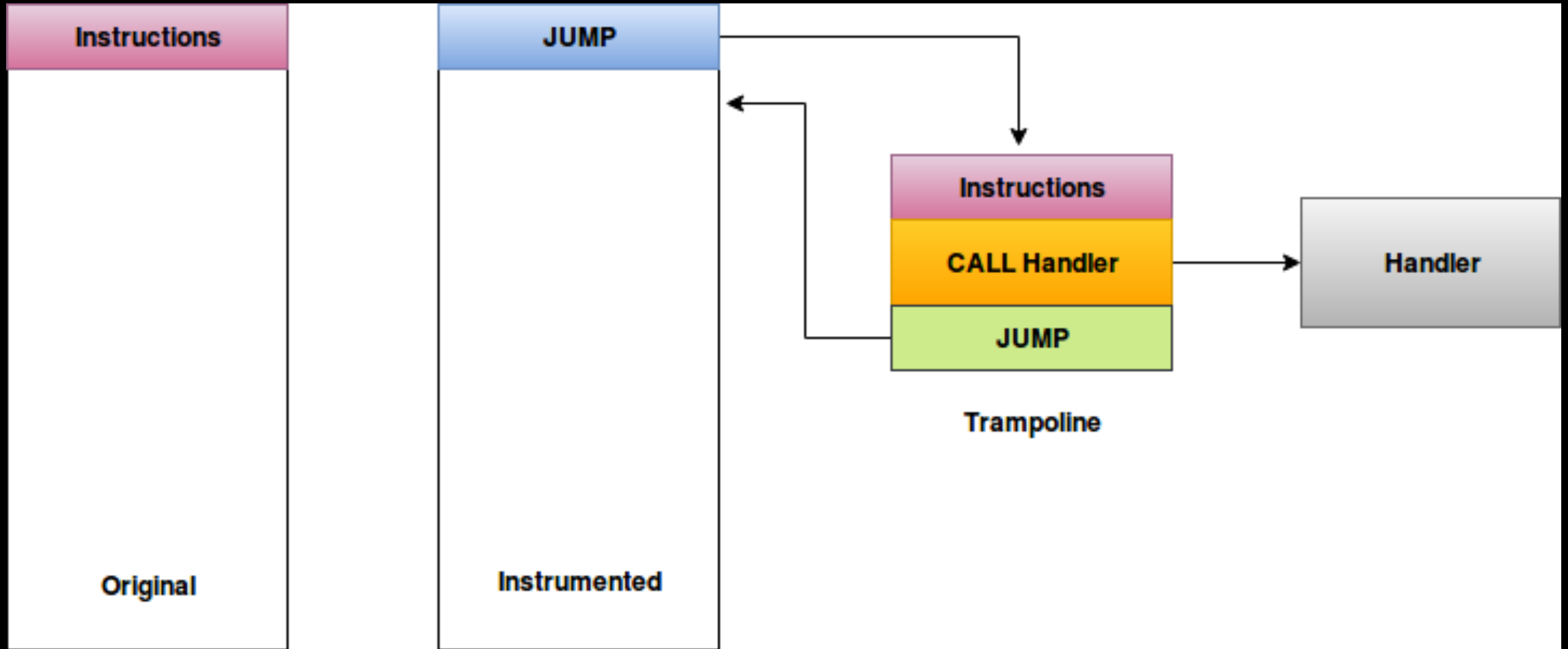
- CFG recovery: correctness v.s. completeness
  - Differentiate data (globals, vftables, jump tables) from code
  - Current effective instrumentation rate: > 60%
  - Rely on IDA Pro to handle compilers & optimizations
- Scalability
  - Tested & works well on Adobe Reader modules (< 10MB - 30MB)
  - For certain compilers, still have FP+ (e.g., mshtml.dll, ~25MB)

# Instrumentation

# Dynamic instrumentation

- Lesson learned from fuzzers based on DBI (Pin/DynamoRio)
  - Unstable & unreliable
  - Limitation on platforms & architectures
  - Poor performance
    - Cannot do selective instrumentation
- Hooking based mechanism
  - Lightweight & selective
  - Offline analysis on where to instrument
    - Handled with static analysis (beforehand)

# Dynamic instrumentation (2)





# SKORPIO instrumentation engine

- Cross-platform: Windows, MacOS, Linux, BSD, etc
- Cross architecture: X86, ARM, ARM64, Mips, PowerPC, Sparc
- Multi-level
  - Userspace & OS kernel
  - Instruction level (vs typical function-entry-only)
- Lightweight
  - Implemented in pure C, focus on low-level hooking mechanism
  - Super fast: can be 100x faster than available public hooking frameworks - thanks to many optimization



## SKORPIO engine (2)

- Decode instructions at hooking place
  - Use Capstone disassembler (X86, ARM, ARM64, Mips, Sparc, PPC, ...)
- Binary rewrite on code relocation
  - Use Keystone assembler (X86, ARM, ARM64, Mips, Sparc, PPC, ...)
- Install user-provided callback at instrumentation hook
- Enable customization/optimization for all requirements
  - Hooking types (JMP or CALL, RET or naked callback)
  - Trampoline setting
  - Thread & internal memory management (OS-agnostic)



# Windows instrumentation

- Inject instrumentation into target binary
  - Instrumentation comes in DLL form
  - DLLMain() runs before main program
- Considered Dynamic DLL injection, but rejected
  - Not portable
- Static inject DLL file into target binary
  - Analyze target PE file to locate Sections & Import Directory
  - Append 1 section to relocate Import Directory
  - Point Import Directory Table to the new appended section
    - Append a new entry for injected DLL

# Linux & MacOS instrumentation

- LD\_PRELOAD to dynamically inject instrumentation
  - Take place before main program runs
  - Linux: shared object file (.so)
  - MacOS: dynamic library (.dylib)
- Inject all instrumentation at initialisation time
  - Can be up to 100k hooks, so must do as quickly as possible
- Inject forkserver at program entry-point, so it takes over later

# Detect heap memory corruption

- Windows
  - Enable PageHeap for fuzzing target
  - Low-level exception handling from Windows core
- MacOS & Linux
  - Built-in memory debugging for better control & performance
    - Overload malloc(), free() & co
    - Utilize MMU to detect overflow/underflow errors
      - Off-by-one error
    - Use-after-free error

**Demo & bugs found**

# Some results

- PoC (Apr 2016): AFL-Cygwin + Intel PIN probe
  - Adobe Reader U3D: 2 unique bugs in 10 hours
    - CVE-2016-1116: Adobe Reader DC U3D e3\_node OOB Access Vulnerability
    - Able to quickly rediscover 12+ bugs on older version:
      - CVE-2014-0523: Adobe Reader U3D Model Node Arbitrary Free Vulnerability
      - CVE-2014-0565: Adobe Reader U3D Line Set Continuation Memory Corruption
      - CVE-2014-9165: Adobe Reader U3D New Object Block Use-after-Free Vulnerability
      - CVE-2015-5586: Adobe Reader U3D Node Blocks Arbitrary Free Memory Corruption
      - CVE-2015-6683: Adobe Reader U3D Bone Weight Modifier Use-after-Free Vulnerability
      - CVE-2016-0933: Adobe Reader DC U3D Bone Weight Modifier OOB Access Vulnerability
      - CVE-2016-1037: Adobe Reader DC U3D Line Set Continuation OOB Access Vulnerability
      - More ... (will release the repros on GitHub)
  - Libxml2-2.7.8.win32 - 10 unique bugs in a week
  - Windows Journal - some bugs

# Experiments

- Libxml2 - native, not compatible with persistent mode
- Native run with persistent mode:
  - UnRAR - persistent mode + parallel fuzzing
  - Msxml6 - persistent mode + parallel fuzzing
  - Adobe Reader - Javascript engine
  - Adobe Reader 3D



# Demos - libxml2.dll

```
C:\Windows\system32\cmd.exe - afl-fuzz.exe -i in -o out -t 1200 -- fuzz-xml.exe @@

GroundX 0.9 <fuzz-xml.exe>

-----
process timing
  run time      : 0 days, 0 hrs, 15 min, 39 sec
  last new path : 0 days, 0 hrs, 0 min, 48 sec
  last uniq crash : 0 days, 0 hrs, 3 min, 33 sec
  last uniq hang  : none seen yet
-----
cycle progress
  now processing : 0 <0.00%>
  paths timed out : 0 <0.00%>
-----
stage progress
  now trying : havoc
  stage execs : 66.9k/80.0k <83.57%>
  total execs : 79.5k
  exec speed  : 82.92/sec
-----
fuzzing strategy yields
  bit flips : 17/544, 7/543, 4/541
  byte flips : 0/68, 2/67, 0/65
  arithmetics : 8/3789, 0/20, 0/0
  known ints  : 4/364, 10/2272, 3/2600
  dictionary  : 0/0, 0/0, 0/0
               havoc : 0/0, 0/0
               trim  : n/a, 0.00%
-----
overall results
  cycles done : 0
  total paths : 172
  uniq crashes : 3
  uniq hangs  : 0
-----
map coverage
  map density : 1933 <2.95%>
  count coverage : 1.00 bits/tuple
-----
findings in depth
  favored paths : 1 <0.58%>
  new edges on  : 172 <100.00%>
  total crashes : 17 (3 unique)
  total hangs  : 0 (0 unique)
-----
path geometry
  levels : 2
  pending : 172
  pend fav : 1
  own finds : 171
  imported : n/a
  variable : 0
```

# Demos - unrar.exe

```
C:\Windows\system32\cmd.exe - afl-fuzz.exe -P 100 -i in2 -M fuzz01 -o sync_dir3 -t 10000+ -f curren... 35
Darko B.93.3 (FuzzB1)
```

<b>process timing</b> run time : 0 days, 0 hrs, 1 min, 1 sec last new path : 0 days, 0 hrs, 0 min, 0 sec last uniq crash : none seen yet last uniq hang : none seen yet	<b>overall results</b> cycles done : 0 total paths : 100 uniq crashes : 0 uniq hangs : 0
<b>cycle progress</b> now processing : 0 (0.00%) paths lined out : 0 (0.00%)	<b>map coverage</b> map density : 3.34% / 5.90% count coverage : 1.00 hits/tuple
<b>stage progress</b> now trying : bitflips 2x1 stage execs : 1100/1839 (60.25%) total execs : 2942 exec speed : 49.00/sec (slow)	<b>findings in depth</b> favored paths : 1 (1.00%) new edges on : 100 (100.00%) total crashes : 0 (0 unique) total ttrouts : 0 (0 unique)
<b>fuzzing strategy yields</b> bit flips : 04/1040, 0/0, 0/0 byte flips : 0/0, 0/0, 0/0 arithmetic : 0/0, 0/0, 0/0 known ints : 0/0, 0/0, 0/0 dictionary : n/a, n/a, n/a haune : 0/0, 0/0 trim : n/a, n/a	<b>path geometry</b> levels : 2 pending : 100 pend fav : 1 own finds : 99 imported : 0 stability : 61.00%

```
C:\Windows\system32\cmd.exe - afl-fuzz.exe -P 100 -i in2 -S fuzz03 -o sync_dir3 -t 10000+ -f curren... 23
Darko B.93.3 (FuzzB3)
```

<b>process timing</b> run time : 0 days, 0 hrs, 0 min, 56 sec last new path : 0 days, 0 hrs, 0 min, 1 sec last uniq crash : none seen yet last uniq hang : none seen yet	<b>overall results</b> cycles done : 0 total paths : 145 uniq crashes : 0 uniq hangs : 0
<b>cycle progress</b> now processing : 21 (14.48%) paths lined out : 0 (0.00%)	<b>map coverage</b> map density : 2.39% / 5.91% count coverage : 1.00 hits/tuple
<b>stage progress</b> now trying : haune stage execs : 684/2000 (35.50%) total execs : 3400 exec speed : 61.00/sec	<b>findings in depth</b> favored paths : 89 (61.38%) new edges on : 145 (100.00%) total crashes : 0 (0 unique) total ttrouts : 0 (0 unique)
<b>fuzzing strategy yields</b> bit flips : n/a, n/a, n/a byte flips : n/a, n/a, n/a arithmetic : n/a, n/a, n/a known ints : n/a, n/a, n/a dictionary : n/a, n/a, n/a haune : 77/863, 21/1520 trim : n/a, n/a	<b>path geometry</b> levels : 0 pending : 141 pend fav : 88 own finds : 105 imported : 39 stability : 51.16%

```
C:\Windows\system32\cmd.exe - afl-fuzz.exe -P 100 -i in2 -S fuzz02 -o sync_dir3 -t 10000+ -f curren... 35
Darko B.93.3 (FuzzB2)
```

<b>process timing</b> run time : 0 days, 0 hrs, 0 min, 58 sec last new path : 0 days, 0 hrs, 0 min, 0 sec last uniq crash : none seen yet last uniq hang : none seen yet	<b>overall results</b> cycles done : 0 total paths : 150 uniq crashes : 0 uniq hangs : 0
<b>cycle progress</b> now processing : 17 (11.33%) paths lined out : 0 (0.00%)	<b>map coverage</b> map density : 1.53% / 5.97% count coverage : 1.00 hits/tuple
<b>stage progress</b> now trying : haune stage execs : 488/3000 (16.27%) total execs : 3496 exec speed : 63.74/sec	<b>findings in depth</b> favored paths : 98 (65.33%) new edges on : 156 (100.00%) total crashes : 0 (0 unique) total ttrouts : 0 (0 unique)
<b>fuzzing strategy yields</b> bit flips : n/a, n/a, n/a byte flips : n/a, n/a, n/a arithmetic : n/a, n/a, n/a known ints : n/a, n/a, n/a dictionary : n/a, n/a, n/a haune : 81/888, 24/1792 trim : n/a, n/a	<b>path geometry</b> levels : 1 pending : 152 pend fav : 94 own finds : 113 imported : 42 stability : 92.43%

```
C:\Windows\system32\cmd.exe - afl-fuzz.exe -P 100 -i in2 -S fuzz04 -o sync_dir3 -t 10000+ -f curren... 23
Darko B.93.3 (FuzzB4)
```

<b>process timing</b> run time : 0 days, 0 hrs, 0 min, 54 sec last new path : 0 days, 0 hrs, 0 min, 0 sec last uniq crash : none seen yet last uniq hang : none seen yet	<b>overall results</b> cycles done : 0 total paths : 142 uniq crashes : 0 uniq hangs : 0
<b>cycle progress</b> now processing : 8 (5.63%) paths lined out : 0 (0.00%)	<b>map coverage</b> map density : 1.29% / 5.90% count coverage : 1.00 hits/tuple
<b>stage progress</b> now trying : splice 1 stage execs : 5/16 (31.25%) total execs : 2958 exec speed : 54.87/sec	<b>findings in depth</b> favored paths : 94 (66.20%) new edges on : 142 (100.00%) total crashes : 0 (0 unique) total ttrouts : 0 (0 unique)
<b>fuzzing strategy yields</b> bit flips : n/a, n/a, n/a byte flips : n/a, n/a, n/a arithmetic : n/a, n/a, n/a known ints : n/a, n/a, n/a dictionary : n/a, n/a, n/a haune : 82/1262, 16/1328 trim : n/a, n/a	<b>path geometry</b> levels : 3 pending : 138 pend fav : 91 own finds : 98 imported : 43 stability : 58.90%

# Demos - afl-tmin.exe

```
λ afl-tmin.exe -t 1000 -i id_000002_11 -o id_000002_11.trim -- fuzz-xml.exe @@ 2>nul
afl-tmin.exe 2.49b by Nguyen Anh Quynh, 2017
Based on AFI 2.49h by <lcantuf@google.com>

[+] Read 68 bytes from 'id_000002_11'.
[*] Performing dry run (timeout = 1000 ms)...
[+] Program terminates normally, minimizing in instrumented mode.
[*] Stage #0: One-time block normalization...
[+] Block normalization complete, 60 bytes replaced.
[*] --- Pass #1 ---
[*] Stage #1: Removing blocks of data...
    Block length = 4, remaining size = 68
    Block length = 2, remaining size = 36
    Block length = 1, remaining size = 36
[+] Block removal complete, 33 bytes deleted.
[*] Stage #2: Minimizing symbols (8 code points)...
[+] Symbol minimization finished, 1 symbol (1 byte) replaced.
[*] Stage #3: Character minimization...
[+] Character minimization done, 0 bytes replaced.
[*] --- Pass #2 ---
[*] Stage #1: Removing blocks of data...
    Block length = 2, remaining size = 35
    Block length = 1, remaining size = 35
[+] Block removal complete, 0 bytes deleted.

    File size reduced by : 48.53% (to 35 bytes)
    Characters simplified : 174.29%
    Number of execs done : 72
        Fruitless execs : path=46 crash=0 hang=0

[*] Writing output to 'id_000002_11.trim'...
[+] We're done here. Have a nice day!
```

# Demos - afl-analyze.exe

```
λ afl-analyze.exe -q -t 1000 -i note.xml -- fuzz-xml.exe @@
afl-analyze.exe 2.49b by Nguyen Anh Quynh, 2017
Based on AFL 2.49b by <lcamtuf@google.com>
```

```
[+] Read 123 bytes from 'note.xml'.
[*] Performing dry run (timeout = 1000 ms)...
[*] Analyzing input file (this may take a while)...
```

01	- no-op block	01	- suspected length field
01	- superficial content	01	- suspected cksum or magic int
01	- critical stream	01	- suspected checksummed block
01	- "magic value" section		

```
[000000] < ? x m l #20 v e r s i o n = " 1
[000016] . @ " > #0d #0a < n o t e > #0d #0a #20 #20
[000032] < t o > T o v e < / t o > #0d #0a #20
[000048] #20 < f r o m > J a n i < / f r o >
[000064] m > #0d #0a #20 #20 < b o d y > D o n '
[000080] t #20 f o r g e t #20 m e #20 t h i s >
[000096] #20 w e e k e n d ! < / b o d y >
[000112] #0d #0a < / n o t e > #0d #0a
```

```
[+] Analysis complete. Interesting bits: 69.92% of the input file.
[+] We're done here. Have a nice day!
```

# Demos - AFL -Q (Linux) vs Darko

american fuzzy lop 2.51b (test2)

process timing		overall results	
run time	: 0 days, 0 hrs, 1 min, 42 sec	cycles done	: 54
last new path	: 0 days, 0 hrs, 1 min, 35 sec	total paths	: 5
last uniq crash	: none seen yet	uniq crashes	: 0
last uniq hang	: none seen yet	uniq hangs	: 0
cycle progress		map coverage	
now processing	: 3 (60.00%)	map density	: 0.06% / 0.10%
paths timed out	: 0 (0.00%)	count coverage	: 1.00 bits/tuple
stage progress		findings in depth	
now trying	: havoc	avored paths	: 5 (100.00%)
stage execs	: 96/256 (37.50%)	new edges on	: 5 (100.00%)
total execs	: 89.1k	total crashes	: 0 (0 unique)
exec speed	: <u>837.8/sec</u>	total tmouts	: 0 (0 unique)
fuzzing strategy yields		path geometry	
bit flips	: 0/80, 0/75, 0/65	levels	: 4
byte flips	: 0/10, 0/5, 0/0	pending	: 0
arithmetics	: 1/560, 0/50, 0/0	pend fav	: 0
known ints	: 0/57, 0/140, 0/0	own finds	: 4
dictionary	: 0/0, 0/0, 0/0	imported	: n/a
havoc	: 3/87.9k, 0/0	stability	: 100.00%
trim	: n/a, 0.00%		

[cpu001:140%]

american fuzzy lop 2.51b (test2)

process timing		overall results	
run time	: 0 days, 0 hrs, 1 min, 45 sec	cycles done	: 138
last new path	: 0 days, 0 hrs, 1 min, 17 sec	total paths	: 7
last uniq crash	: 0 days, 0 hrs, 1 min, 17 sec	uniq crashes	: <b>1</b>
last uniq hang	: none seen yet	uniq hangs	: 0
cycle progress		map coverage	
now processing	: 1 (14.29%)	map density	: 0.02% / 0.04%
paths timed out	: 0 (0.00%)	count coverage	: 1.00 bits/tuple
stage progress		findings in depth	
now trying	: havoc	avored paths	: 7 (100.00%)
stage execs	: 195/256 (76.17%)	new edges on	: 7 (100.00%)
total execs	: 371k	total crashes	: <b>120 (1 unique)</b>
exec speed	: <u>3272/sec</u>	total tmouts	: 0 (0 unique)
fuzzing strategy yields		path geometry	
bit flips	: 0/136, 0/129, 0/115	levels	: 6
byte flips	: 0/17, 0/10, 0/1	pending	: 0
arithmetics	: 2/952, 0/75, 0/0	pend fav	: 0
known ints	: 0/95, 0/280, 0/44	own finds	: 6
dictionary	: 0/0, 0/0, 0/0	imported	: n/a
havoc	: 5/368k, 0/0	stability	: 100.00%
trim	: n/a, 0.00%		

[cpu000:137%]

# Demos - Darko vs AFL native on MacOS

american fuzzy lop 2.51b (test1)

## process timing

run time : 0 days, 0 hrs, 1 min, 37 sec  
last new path : 0 days, 0 hrs, 1 min, 16 sec  
last uniq crash : 0 days, 0 hrs, 1 min, 30 sec  
last uniq hang : none seen yet

## cycle progress

now processing : 6 (85.71%)  
paths timed out : 0 (0.00%)

## stage progress

now trying : havoc  
stage execs : 40/512 (7.81%)  
total execs : 102k  
exec speed : 1010/sec

## fuzzing strategy yields

bit flips : 0/144, 0/137, 1/123  
byte flips : 0/18, 0/11, 0/2  
arithmetics : 2/1008, 0/74, 0/0  
known ints : 0/101, 0/308, 0/88  
dictionary : 0/0, 0/0, 0/0  
havoc : 4/90.0k, 0/10.5k  
trim : 55.56%/2, 0.00%

## overall results

cycles done : **21**  
total paths : 7  
uniq crashes : **1**  
uniq hangs : 0

## map coverage

map density : **0.00% / 0.07%**  
count coverage : 1.00 bits/tuple

## findings in depth

favored paths : 7 (100.00%)  
new edges on : 7 (100.00%)  
total crashes : **12 (1 unique)**  
total tmouts : 0 (0 unique)

## path geometry

levels : 5  
pending : 0  
pend fav : 0  
own finds : 6  
imported : n/a  
stability : 100.00%

[cpu: 93%]

american fuzzy lop 2.51b (test2)

## process timing

run time : 0 days, 0 hrs, 1 min, 35 sec  
last new path : 0 days, 0 hrs, 0 min, 32 sec  
last uniq crash : 0 days, 0 hrs, 0 min, 17 sec  
last uniq hang : none seen yet

## cycle progress

now processing : 6 (85.71%)  
paths timed out : 0 (0.00%)

## stage progress

now trying : havoc  
stage execs : 736/2048 (35.94%)  
total execs : 115k  
exec speed : 1158/sec

## fuzzing strategy yields

bit flips : 0/136, 0/129, 0/115  
byte flips : 0/17, 0/10, 0/1  
arithmetics : 2/952, 0/50, 0/0  
known ints : 0/96, 0/280, 0/44  
dictionary : 0/0, 0/0, 0/0  
havoc : 5/112k, 0/0  
trim : 20.00%/1, 0.00%

## overall results

cycles done : **81**  
total paths : 7  
uniq crashes : **1**  
uniq hangs : 0

## map coverage

map density : **0.02% / 0.04%**  
count coverage : 1.00 bits/tuple

## findings in depth

favored paths : 7 (100.00%)  
new edges on : 7 (100.00%)  
total crashes : **14 (1 unique)**  
total tmouts : 0 (0 unique)

## path geometry

levels : 5  
pending : 0  
pend fav : 0  
own finds : 6  
imported : n/a  
stability : 100.00%

[cpu: 93%]

# Demos - MacOS

**american fuzzy top 2.51b (0)**

<b>process timing</b>		<b>overall results</b>	
run time : 0 days, 0 hrs, 1 min, 58 sec	cycles done : 0	total paths : 612	
last new path : 0 days, 0 hrs, 0 min, 3 sec	unq crashes : 0	unq hangs : 0	
last unq crash : none seen yet			
last unq hang : none seen yet			
<b>cycle progress</b>		<b>map coverage</b>	
now processing : 0 (0.78%)	map density : 1.28K / 4.58K	count coverage : 2.23 bits/tuple	
paths timed out : 0 (0.000)	findings in depth		
<b>stage progress</b>		<b>findings in depth</b>	
now trying : with 0/3	favored paths : 275 (30.000)	new edges on : 393 (46.180)	
stage execs : 1546/24.2k (6.370)	total crashes : 0 (0 unique)	total hangs : 0 (0 unique)	
total execs : 62.4k			
exec speed : 506.2/sec			
<b>fuzzing strategy yields</b>		<b>path geometry</b>	
bit flips : 21/3696, 3/3654, 4/3658	levels : 2	pending : 314	
byte flips : 0/417, 2/415, 0/464	pend fav : 275	new finds : 92	
arithmetics : 7/5805, 0/283, 0/0	unq finds : 92	imported : 375	
known ints : 4/599, 0/2516, 0/4932	stability : 100.000		
dictionary : 0/0, 0/0, 4/499			
havoc : 45/24.8k, 0/0			
trim : 3.380/602, 0.000			

[cpu: 300%]

**american fuzzy top 2.51b (52)**

<b>process timing</b>		<b>overall results</b>	
run time : 0 days, 0 hrs, 1 min, 23 sec	cycles done : 0	total paths : 603	
last new path : 0 days, 0 hrs, 0 min, 1 sec	unq crashes : 0	unq hangs : 0	
last unq crash : none seen yet			
last unq hang : none seen yet			
<b>cycle progress</b>		<b>map coverage</b>	
now processing : 149 (16.168%)	map density : 0.78K / 4.60K	count coverage : 2.23 bits/tuple	
paths timed out : 0 (0.000)	findings in depth		
<b>stage progress</b>		<b>findings in depth</b>	
now trying : splice 15	favored paths : 285 (30.360)	new edges on : 397 (44.460)	
stage execs : 10/32 (31.250)	total crashes : 0 (0 unique)	total hangs : 0 (0 unique)	
total execs : 41.1k			
exec speed : 515.4/sec			
<b>fuzzing strategy yields</b>		<b>path geometry</b>	
bit flips : n/a, n/a, n/a	levels : 3	pending : 899	
byte flips : n/a, n/a, n/a	pend fav : 299	new finds : 73	
arithmetics : n/a, n/a, n/a	imported : 228	stability : 99.30%	
known ints : n/a, n/a, n/a			
dictionary : n/a, n/a, n/a			
havoc : 3/932, 76/21.6k			
trim : 42.130/760, n/a			

[cpu: 300%]

**american fuzzy top 2.51b (51)**

<b>process timing</b>		<b>overall results</b>	
run time : 0 days, 0 hrs, 1 min, 34 sec	cycles done : 0	total paths : 600	
last new path : 0 days, 0 hrs, 0 min, 0 sec	unq crashes : 0	unq hangs : 0	
last unq crash : none seen yet			
last unq hang : none seen yet			
<b>cycle progress</b>		<b>map coverage</b>	
now processing : 130 (14.40%)	map density : 0.60K / 4.51K	count coverage : 2.23 bits/tuple	
paths timed out : 0 (0.000)	findings in depth		
<b>stage progress</b>		<b>findings in depth</b>	
now trying : havoc	favored paths : 279 (31.000)	new edges on : 415 (46.110)	
stage execs : 320/532 (52.500)	total crashes : 0 (0 unique)	total hangs : 0 (0 unique)	
total execs : 47.4k			
exec speed : 519.4/sec			
<b>fuzzing strategy yields</b>		<b>path geometry</b>	
bit flips : n/a, n/a, n/a	levels : 3	pending : 373	
byte flips : n/a, n/a, n/a	pend fav : 254	new finds : 95	
arithmetics : n/a, n/a, n/a	imported : 148	stability : 100.000	
known ints : n/a, n/a, n/a			
dictionary : n/a, n/a, n/a			
havoc : 36/21.8k, 58/15.8k			
trim : 46.210/613, n/a			

[cpu: 300%]

**american fuzzy top 2.51b (53)**

<b>process timing</b>		<b>overall results</b>	
run time : 0 days, 0 hrs, 1 min, 11 sec	cycles done : 0	total paths : 654	
last new path : 0 days, 0 hrs, 0 min, 0 sec	unq crashes : 0	unq hangs : 0	
last unq crash : none seen yet			
last unq hang : none seen yet			
<b>cycle progress</b>		<b>map coverage</b>	
now processing : 114 (16.700%)	map density : 0.61K / 4.63K	count coverage : 2.22 bits/tuple	
paths timed out : 0 (0.000)	findings in depth		
<b>stage progress</b>		<b>findings in depth</b>	
now trying : splice 1	favored paths : 281 (32.730)	new edges on : 400 (46.620)	
stage execs : 31/32 (96.880)	total crashes : 0 (0 unique)	total hangs : 0 (0 unique)	
total execs : 76.6k			
exec speed : 518.2/sec			
<b>fuzzing strategy yields</b>		<b>path geometry</b>	
bit flips : n/a, n/a, n/a	levels : 3	pending : 633	
byte flips : n/a, n/a, n/a	pend fav : 258	new finds : 32	
arithmetics : n/a, n/a, n/a	imported : 285	stability : 99.30%	
known ints : n/a, n/a, n/a			
dictionary : n/a, n/a, n/a			
havoc : 10/12.7k, 22/13.4k			
trim : 46.310/541, n/a			

[cpu: 300%]

[0] @ afl-fuzz\*

"C:\fuzz\local" 18:27 03-Nov-15

# Conclusions

- **DARKO** is an advanced coverage-guided fuzzer
  - Pure software-based
  - Cross-platform/architecture
  - Binary support
    - Fuzz full binary + Persistent mode
  - Fast + stable
- **SKORPIO** engine will be released to public in near future



# Questions?

## Building Advanced Coverage-guided Fuzzer for Program Binaries

NGUYEN Anh Quynh <[aquynh@gmail.com](mailto:aquynh@gmail.com)>

WEI Lei