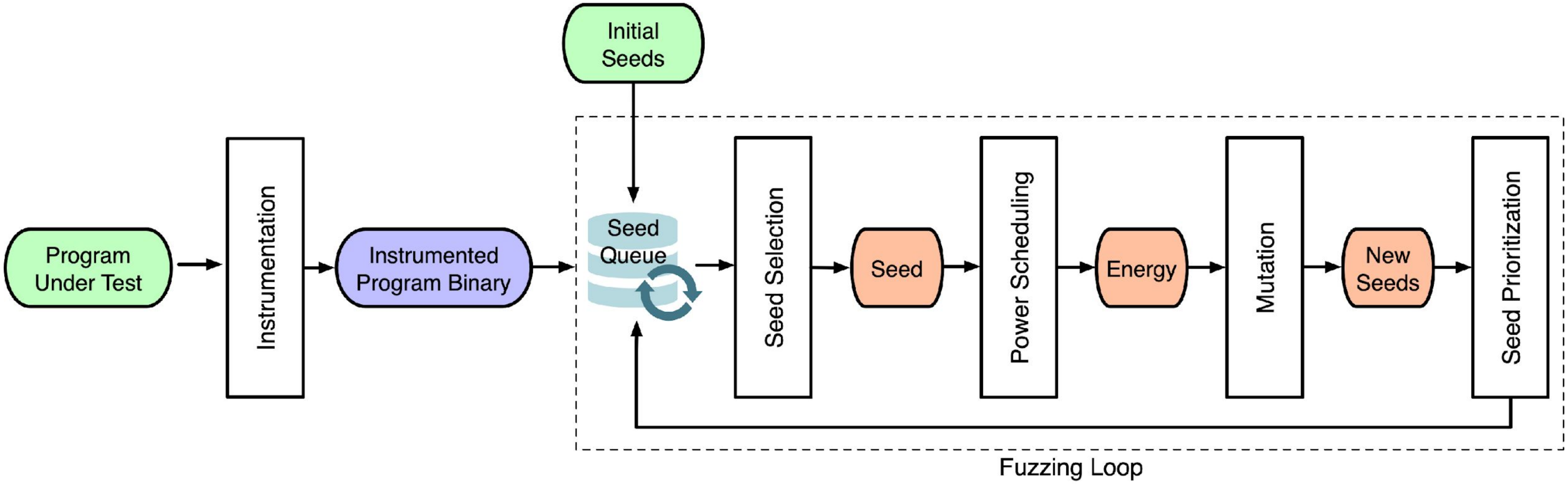
Hawkeye: Towards a Desired Directed Grey-box Fuzzing

Hongxu Chen, Yinxing Xue, Yuekang Li,
Bihuan Chen, Xiaofei Xie, Xiuheng Wu, Yang Liu

October 18, 2018

Mutation Based Grey-box Fuzzing



- General-purpose Grey-box Fuzzing: Cover more paths and induce more bugs (if any)
- Directed Grey-box Fuzzing (DGF): Given a target site (e.g., file & line number), test this site intensively, and induce more relevant bugs

Why Directed Grey-box Fuzzing ? (1)

```
diff --git a/bfd/dwarf2.c b/bfd/dwarf2.c
index 1566cd8..8abb3f0 100644 (file)
--- a/bfd/dwarf2.c
+++ b/bfd/dwarf2.c
@@ -1933,6 +1933,13 @@ read_formatted_entries (struct comp_unit *unit, bfd_byte **bufp,

   data_count = _bfd_safe_read_leb128 (abfd, buf, &bytes_read, FALSE, buf_end);
   buf += bytes_read;
+  if (format_count == 0 && data_count != 0)
+    {
+      _bfd_error_handler (_("Dwarf Error: Zero format count."));
+      bfd_set_error (bfd_error_bad_value);
+      return FALSE;
+    }
+
   for (datai = 0; datai < data_count; datai++)
     {
       bfd_byte *format = format_header_data;
```

Patch Testing

Why Directed Grey-box Fuzzing ? (2)

Project Name	CID	Checker	Category
wazuh/ossec-wazuh	117766	USE_AFTER_FREE	Memory - illegal accesses

File: /wazuh_modules/wmodules.c

```
< 4. Condition "cur_module", taking true branch
57      for (cur_module = wmodules; cur_module; wmodules = next_module) {
<<< CID 117766: Memory - illegal accesses USE_AFTER_FREE
<<< 5. Dereferencing freed pointer "cur_module".
58          next_module = cur_module->next;
59          cur_module->context->destroy(cur_module->data);
<< 2. "free" frees "cur_module".
60          free(cur_module);
< 3. Jumping back to the beginning of the loop
```

Justify a suspicious vulnerability

Why Directed Grey-box Fuzzing ? (3)

CVE-2016-1835 Detail

MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

Current Description

Use-after-free vulnerability in the xmlSAX2AttributeNs function in libxml2 before 2.9.4, as used in Apple iOS before 9.3.2 and OS X before 10.11.5, allows remote attackers to cause a denial of service via a crafted XML document.

Source: MITRE

Description Last Modified: 07/27/2016

+View Analysis Description

Crash Reproduction based on vulnerability description

Desired Properties for DGF (1)

P1: A distance metric avoiding bias to certain traces reachable to targets

- ➢ *All traces* reachable to the target should be considered
- ➢ e.g., Given a patch for GNU Binutils nm CVE-2017-15023, there are >=2 traces reachable to **dwarf2.c:1601** in ***concat_filename***

Functions in a Crashing Trace	File & Line	Symbol
main	nm.c :1794	M
...	...	...
_bfd_dwarf2_find_nearest_line	dwarf2.c :4798	a
comp_unit_find_line	dwarf2.c :3686	b
comp_unit_maybe_decode_line_info	dwarf2.c :3651	c
decode_line_info	dwarf2.c :2265	d
concat_filename	dwarf2.c :1601	T
...	...	Z
Functions in a Normal Trace	**File & Line**	**Symbol**
main	nm.c :1794	M
...	...	...
_bfd_dwarf2_find_nearest_line	dwarf2.c :4798	a
scan_unit_for_symbols	dwarf2.c :3211	e
concat_filename	dwarf2.c :1601	T
...	...	Z

Desired Properties for DGF (2)

P2: Balance cost-effectiveness between static analysis and dynamic analysis

1. static analysis ***has to*** be applied for DGF
2. Precise static analysis ***can be costly*** but ***may not be useful*** for dynamic fuzzing
3. Coarse static analysis provides ***little directedness*** for fuzzing

Desired Properties for DGF (3)

P3: Prioritize proper seeds and schedule mutations

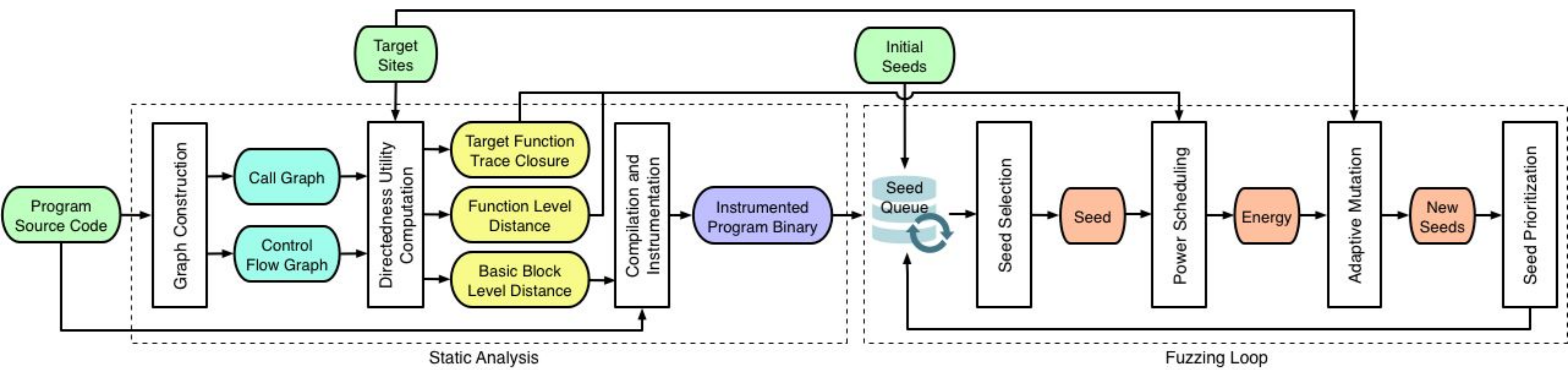
- Prioritization can boost DGF significantly
  - variants of certain seeds have less chances to reach the target sites
  - some seeds contribute little in exploring new execution traces
- Scheduling more mutations on “good” seeds are more beneficial

Desired Properties for DGF (4)

P4: Adaptive mutation to increase mutators' effectiveness

- Coarse-grained mutations typically change the execution traces greatly
- Apply more fine-grained mutations when execution traces are close to the target sites

Overall Workflow of Hawkeye



PART 1: Static Analysis

- **Compute static distance utilities**
  a. Apply whole program analysis to construct Interprocedural Control Flow Graph (ICFG)
  b. Build static directedness utilities w.r.t. target site(s) based on ICFG
  c. Instrument directedness utilities into the program under test

Graph Construction

1. Call Graph (CG)
   a. Andersen's pointer analysis
   b. Function pointers ⇒ Indirect calls
      i. Much more precise than explicit-only Call Graph
      ii. Less costly than context-/flow-sensitive analysis
2. Control Flow Graph (CFG)
3. CG + CFG ⇒ ICFG

Adjacent-Function Distance Augmentation (1)

```
void fa(int i) {
  if (i > 0) {
    fb(i);
  } else {
    fb(i*2);
    fc();
  }
}
```

```
void fa(int i) {
  if (i > 0) {
    fb(i);
    fb(i*2);
  } else {
    fc();
  }
}
```

How to determine the distances of fa→ fb and fa → fc ?

Adjacent-Function Distance Augmentation (2)

f_1: Caller f_2: callee

C_N: Call sites occurrences of f_2 inside f_1

C_B: No. of basic blocks in f_1 that contains >= *1* call site of f_2

$$d_f(f_1, f_2) = \frac{\phi \cdot C_N + 1}{\phi \cdot C_N} \cdot \frac{\psi \cdot C_B + 1}{\psi \cdot C_B}$$

Adjacent-Function Distance Augmentation (3)

```
void fa(int i) {
  if (i > 0) {
    fb(i);
  } else {
    fb(i*2);
    fc();
  }
}
```

```
void fa(int i) {
  if (i > 0) {
    fb(i);
    fb(i*2);
  } else {
    fc();
  }
}
```

Let $\phi = 2$ and $\psi = 2$,

$$d_f(f_a, f_b) = \frac{2 \cdot 2 + 1}{2 \cdot 2} \cdot \frac{2 \cdot 2 + 1}{2 \cdot 2} = 1.56$$

$$d_f(f_a, f_c) = \frac{2 \cdot 1 + 1}{2 \cdot 1} \cdot \frac{2 \cdot 1 + 1}{2 \cdot 1} = 2.25$$

$$d'_f(f_a, f_b) = \frac{2 \cdot 2 + 1}{2 \cdot 2} \cdot \frac{2 \cdot 1 + 1}{2 \cdot 1} = 1.87$$

$$d'_f(f_a, f_c) = \frac{2 \cdot 1 + 1}{2 \cdot 1} \cdot \frac{2 \cdot 1 + 1}{2 \cdot 1} = 2.25$$

Directedness Utility Computation

- ***$d_f(f_s, f_t)$***: distance between *any two* functions ***f_s*** and ***f_t*** in the call graph
- ***$d_f(n, T_f)$***: function level distance to target(s), where ***n*** is a function, ***T_f*** is the set of target functions
- ***$d_b(m, T_b)$***: basic block distance to target(s)
- ***$\xi_f(T_f)$***: target function trace closure

PART 2: Fuzzing Loop

- **Dynamic fuzzing based on static utilities and feedback**
  - Track two separate execution metrics to measure "distance" between current trace and "expected" traces
  - Calculate a power function based on the two metrics
  - Schedule mutation chances based on power function
  - Adaptively mutate based on reachability to target sites
  - Prioritize seeds based on power function and coverage

Two Metrics

Basic Block Trace Distance:

$$d_s(s, T_b) = \frac{\Sigma_{m \in \xi_b(s)} d_b(m, T_b)}{|\xi_b(s)|}$$

Covered Function Similarity:

$$c_s(s, T_f) = \frac{\Sigma_{f \in \xi_f(s) \cap \xi_f(T_f)} d_f(f, T_f)^{-1}}{|\xi_f(s) \cup \xi_f(T_f)|}$$

Power Function

$$p(s, T_b) = c_s(s, T_f) \cdot (1 - \tilde{d}_s(s, T_b))$$

- C_s favors longer traces that share more executed functions with the “expected” traces
- d_s favors shorter traces that reach the expected targets
- Used directly for scheduling mutation chances

Adaptive Mutation

When a seed has reached target functions, prefer fine-grained mutations

- Fine-grained: bit/byte level flips, add/sub on bytes/words, replace with interesting values
- Coarse-grained: random chunk modifications, semantic mutations, crossover

Seed Prioritization

A *three-tier* queue to differentiate seed priorities and favor seeds that:

a. cover new edges
b. are close to targets
c. reach target function(s)

Hawkeye's Solution to Desired Properties

P1: Combine basic block trace distance and covered function similarity for power function to avoid bias

P2: Apply precise graph construction and argument adjacent-function distance to generate cost-effective directedness utilities for dynamic fuzzing

P3: Apply target-favored seed prioritization and mutation power scheduling

P4: Apply adaptive mutation based on reachability to targets

Evaluation Tools

- **Hawkeye**: Our proposed fuzzer that tries to satisfy the proposed four desired properties
- **Fidgety-AFL**: State-of-the-art coverage-oriented Grey-box fuzzer
- **AFLGo**: DGF based on basic block distance instrumentation and simulated annealing scheduling
- **HE-Go**: DGF whose basic block distance instrumentation follows Hawkeye’s, but uses AFLGo’s scheduling

Crash Reproduction (cxxfilt)

CVE-ID	Tool	Runs	μTTE(s)	Factor
2016-4487 2016-4488	Hawkeye	20	177	–
	AFLGo	20	390	2.20
	AFL	20	630	3.56
2016-4489	Hawkeye	20	206	–
	AFLGo	20	180	0.87
	AFL	20	420	2.04
2016-4490	Hawkeye	20	103	–
	AFLGo	20	93	0.90
	AFL	20	59	0.57
2016-4491	Hawkeye	9	18733	–
	AFLGo	5	23880	1.27
	AFL	7	20760	1.11
2016-4492 2016-4493	Hawkeye	20	477	–
	AFLGo	20	540	1.21
	AFL	20	960	2.01
2016-6131	Hawkeye	9	17314	–
	AFLGo	6	21180	1.22
	AFL	2	26340	1.52

Crash Reproduction (MJS)

Bug ID	Tool	Runs	μTTE(s)	Factor	A_{12}
#1	Hawkeye	5	5469	–	–
	AFLGo	2	12581	2.30	0.77
	AFL	2	13084	2.39	0.77
#2	Hawkeye	7	1880	–	–
	AFLGo	2	12753	6.78	0.95
	AFL	2	12294	6.54	0.95
#3	Hawkeye	8	178	–	–
	AFLGo	8	819	4.60	0.91
	AFL	8	1269	7.13	0.95
#4	Hawkeye	8	5519	–	–
	AFLGo	8	5878	1.07	0.57
	AFL	8	5036	0.91	0.48

#1 Stack Overflow
#2 Invalid read
#3 Heap buffer overflow
#4 Use after free

Crash Reproduction (Oniguruma)

Bug ID	Tool	Runs	μTTE(s)	Factor	A_{12}
#1	Hawkeye	8	139	–	–
	HE-Go	8	149	1.07	0.58
	AFL	8	135	0.97	0.54
#2	Hawkeye	8	186	–	–
	HE-Go	8	228	1.23	0.88
	AFL	8	372	2.00	1.0
#3	Hawkeye	2	13768	–	–
	HE-Go	1	14163	1.03	0.56
	AFL	1	14341	1.04	0.57
#4	Hawkeye	7	6969	–	–
	HE-Go	3	12547	1.80	0.82
	AFL	1	14375	2.06	0.88

#1, #2, #3 are from Oniguruma 6.2.0
#4 is from Oniguruma 6.8.2

Target Site Covering (Google Fuzzer Test Suite)

ID	Project	Tool	Runs	μTTE(s)	Factor	A_{12}
#1	jdmarker.c:659	Hawkeye	8	1955	–	–
		HE-Go	8	2012	1.03	0.53
		AFL	8	4839	2.48	0.95
#2	pngread.c:738	Hawkeye	8	23	–	–
		HE-Go	8	16	0.70	0.43
		AFL	8	130	5.65	1.00
#3	pngrutil.c:3182	Hawkeye	8	1	–	–
		HE-Go	8	66	66.00	0.56
		AFL	8	3	3.00	0.51
#4	ttgload.c:1710	Hawkeye	7	4283	–	–
		HE-Go	7	4443	1.04	0.55
		AFL	6	5980	1.40	0.60

Summary

1. Directed Grey-box Fuzzing (DGF) can be helpful
2. We analyzed the challenges in DGF and developed a fuzzer Hawkeye aiming to satisfy the desired properties
3. Experimental results demonstrate Hawkeye's effectiveness in both crash reproduction and target site covering

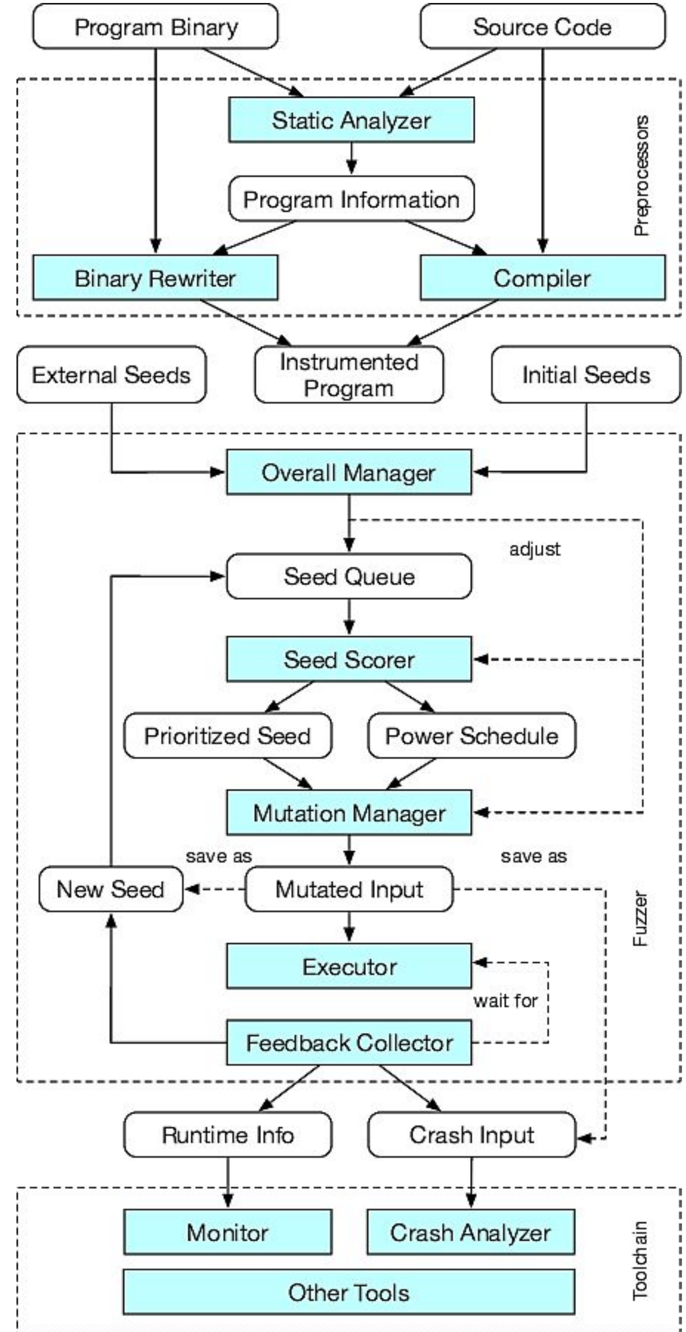
FOT: A Versatile, Configurable, Extensible Fuzzing Framework (Fuzzing Orchestration Toolkit)

- highly modularized
- supports different features

Features \ Framework	AFL	libFuzzer	honggfuzz	FOT
Binary-Fuzzing Support	●	○	●	●
Multi-threading Mode	○	●	●	●
In-memory Fuzzing	●	●	●	●
Advanced Configuration	○	◐	○	●
Modularized Functionality	○	◐	○	●
Structure-aware Mutation	○	○	○	◐
Interoperability	○	○	○	◐
Toolchain Support	●	○	○	●
Precise Crash Analysis	○	○	●	●
Runtime Visualization	◐	○	○	●

See our upcoming ESEC/FSE18

Demo: https://bit.ly/2yzLFla



Thank you !

Two Relevant CVEs in Binutils nm (NULL pointer Read)

$ nm -A -a -l -S -s --special-syms --synthetic --with-symbol-versions -D $POC1
==3765==ERROR: AddressSanitizer: SEGV on unknown address 0x000000000000
==3765==The signal is caused by a READ memory access.
==3765==Hint: address points to the zero page.
    #0 0x6a7375 in **concat_filename**
/home/hawkeye/binutils/bfd/dwarf2.c:1601:8
    #1 0x696e83 in **decode_line_info**
/home/hawkeye/binutils/bfd/dwarf2.c:2258:44
    #2 0x6a2ab8 in **comp_unit_maybe_decode_line_info**
/home/hawkeye/binutils/bfd/dwarf2.c:3642:26
    #3 0x6a2ab8 in **comp_unit_find_line**
/home/hawkeye/binutils/bfd/dwarf2.c:3677
    #4 0x6a0104 in _bfd_dwarf2_find_nearest_line
/home/hawkeye/binutils/bfd/dwarf2.c:4789:11
    #5 0x5f330e in **_bfd_elf_find_line** /home/hawkeye/binutils/bfd/elf.c:8695:10
    #6 0x5176a3 in **print_symbol** /home/hawkeye/binutils/binutils/nm.c:1003:9
    #7 0x514e4d in **print_symbols** /home/hawkeye/binutils/binutils/nm.c:1084:7
    #8 0x514e4d in **display_rel_file** /home/hawkeye/binutils/binutils/nm.c:1200
    #9 0x510976 in **display_file** /home/hawkeye/binutils/binutils/nm.c:1318:7
    #10 0x50f4ce in **main** /home/hawkeye/binutils/binutils/nm.c:1792:12

CVE-2017-15023

$ nm -A -a -l -S -s --special-syms --synthetic --with-symbol-versions -D $POC2
==19042==ERROR: AddressSanitizer: SEGV on unknown address
0x000000000000
==19042==The signal is caused by a READ memory access.
==19042==Hint: address points to the zero page.
    #0 0x6a76a5 in **concat_filename**
/home/hawkeye/binutils/bfd/dwarf2.c:1601:8
    #1 0x696ff3 in **decode_line_info**
/home/hawkeye/binutils/bfd/dwarf2.c:2265:44
    #2 0x6a2d36 in **comp_unit_maybe_decode_line_info**
/home/hawkeye/binutils/bfd/dwarf2.c:3651:26
    #3 0x6a2d36 in **comp_unit_find_line**
/home/hawkeye/binutils/bfd/dwarf2.c:3686
    #4 0x6a0369 in _bfd_dwarf2_find_nearest_line
/home/hawkeye/binutils/bfd/dwarf2.c:4798:11
    #5 0x5f332e in **_bfd_elf_find_line** /home/hawkeye/binutils/bfd/elf.c:8695:10
    #6 0x5176a3 in **print_symbol** /home/hawkeye/binutils/binutils/nm.c:1003:9
    #7 0x514e4d in **print_symbols** /home/hawkeye/binutils/binutils/nm.c:1084:7
    #8 0x514e4d in **display_rel_file** /home/hawkeye/binutils/binutils/nm.c:1200
    #9 0x510976 in **display_file** /home/hawkeye/binutils/binutils/nm.c:1318:7
    #10 0x50f4ce in **main** /home/hawkeye/binutils/binutils/nm.c:1792:12

CVE-2017-15939

Statistics of Tested Programs

Project	Program	Size	ics	cs	ics/cs	# of C_B>1	# of C_N>1	t_s
Binutils	cxxfilt	2.8M	3232	12117	26.67%	8813	8879	735s
Oniguruma	testcu	1.3M	556	2065	26.93%	3037	3101	5s
mjs	mjs	277K	130	3277	3.97%	309	334	3s
libjpeg	libjpeg	810K	749	1827	41.00%	144	152	2s
libpng	libpng	228K	449	1018	44.11%	61	61	2s
freetype2	freetype	1.6M	627	5681	11.30%	6784	7117	4s

Selected Trophies

- binaryen: 17 bugs
- CImg: 2 bugs
- Espruino: 9 CVEs
- FFmpeg: 3 CVEs
- FLIF: 2 bugs
- GNU bc: 18 bugs
- GNU Binutils: 1 CVE
- GNU diffutils: 2 bugs
- GPAC: 15 bugs
- imagemagick: 2 CVEs
- Intel XED: 2 bugs
- libjpeg-turbo: 1 CVE
- liblouis: 1 CVE
- lepton: 4 bugs
- libsass: 10 bugs
- libvips: 11 bugs
- Oniguruma: 6 CVEs
- radare2: 40+ bugs
- MJS: 33 bugs
- Swift: 7 bugs