Review ‘Ddisasm’

Name: Wenhui Zhang
Email: wuz49@psu.edu

Paper Name:
Flores-Montoya, A., & Schulte, E. (2020). Datalog disassembly. In *29th {USENIX} Security Symposium ({USENIX} Security 20)* (pp. 1075-1092).

Contribution:
This paper proposes and implements a binary disassembly and reassembly framework in datalog named Ddisasm. Ddisasm parses ELF files and translates the binary to a symbolic representation exposing control-flow and data-flow dependencies.

Motivation:
To patch the vulnerable programs and fix the bugs, source code with assembly level or above is needed for the instrumentations. However, source code is not always available for these programs. Thus, there requires a way for translating binary code back into source code with assembly level or above. Re-assembleable assembly is needed for binary analysis and binary rewriting and hardening. Previous works of binary disassembly and reassembly are based on linear sweep and recursive traversal approaches. Linear sweep may lead to inaccurate translations, since it may mismatch of databytes and assembly instructions. Recursive traversal approach is complex and is not efficient.

Methodology:
Ddisasm generates assembly code with cross references,and assists with modification of ELF without breaking it in two steps: (1) Instruction Boundary Identification, which amounts to finding which addresses correspond to which instructions; (2) Symbolization, which distinguishes numbers from references to literals.

Results:

Benchmark	Programs	Binaries	References	Ddisasm					Ramblr				
				FP	FN	Wrong-Sect	Broken	Broken%	FP	FN	Wrong-Sect	Broken	Broken%
Real world	25	600	3446843	1	14	36	10	1.66%	1076	429	-	213	35.50%
Coreutils	106	2120	2381924	0	0	0	0	0%	13	21	-	33	1.55%
CGC	69	1656	4531345	0	3	0	3	0.18%	56	68	-	82	4.96%

TABLE I. SYMBOLIZATION EVALUATION. Evaluation of symbolization information recovered by `Ddisasm` and `Ramblr` from 4376 binaries. Recovered symbolization information is compared to ground truth extracted from binaries generated with full relocation information. The "Benchmark" column lists the class of benchmark programs; "Programs" lists the number of programs in this benchmark class; "Binaries" lists the total number of binaries compiled from these programs using multiple compilers and optimization flags; "Refs" represents the ground truth total number of references in these binaries; "FP" and

Ddisasm is tested with 3 benchmark sets, 7 compiler versions (GCC, Clang and ICC) and 6 compiler optimization flags. In all these tests, Ddisasm is both faster and more accurate than the current state-of-the-art binary reassembling tool, Ramblr, on a total of 7,658 ELF Linux x64 binaries.

Discussion and Take away:

1. Ddisasm's approaches rely on practical heuristics, where it holds assumptions about what common compilers and common assembler code is doing. However, most released binaries today are packed by program packers. The binaries are obfuscated, which breaks heuristics on the code which is built up by common assemblers/compilers. One thing we could possibly do is to add heuristics of common packers to the datalog file, and while conducting disassembling and assembling, use this datalog for assisting the approach.
2. Sometimes the datalog might be mis-interpreted or mis-presented. One thing we could possibly do is to add Satisfiability modulo theories (SMT) on top of datalog to ensure its correctness.