BinForce

Statically Rewriting x86 COTS Binaries w/o Heuristics

Erick Bauman and Zhiqiang Lin

University of Texas at Dallas

Static Binary Rewriting is Important

Applications

1. Software fault isolation (SFI) [WLAG93]
2. Control Flow Integrity (CFI) [ABEL09]
3. Binary code hardening (e.g., STIR [WMHL12])
4. Binary code reuse (e.g., BCR [CJMS10])
5. Platform-specific optimizations [ASE+13]

Challenges in Disassembling

1. Recognizing and relocating static memory addresses
2. Handling dynamically computed memory addresses
3. Differentiating code from data
4. Handling function pointer arguments (e.g., callbacks)
5. Handing PIC (Position Independent Code)

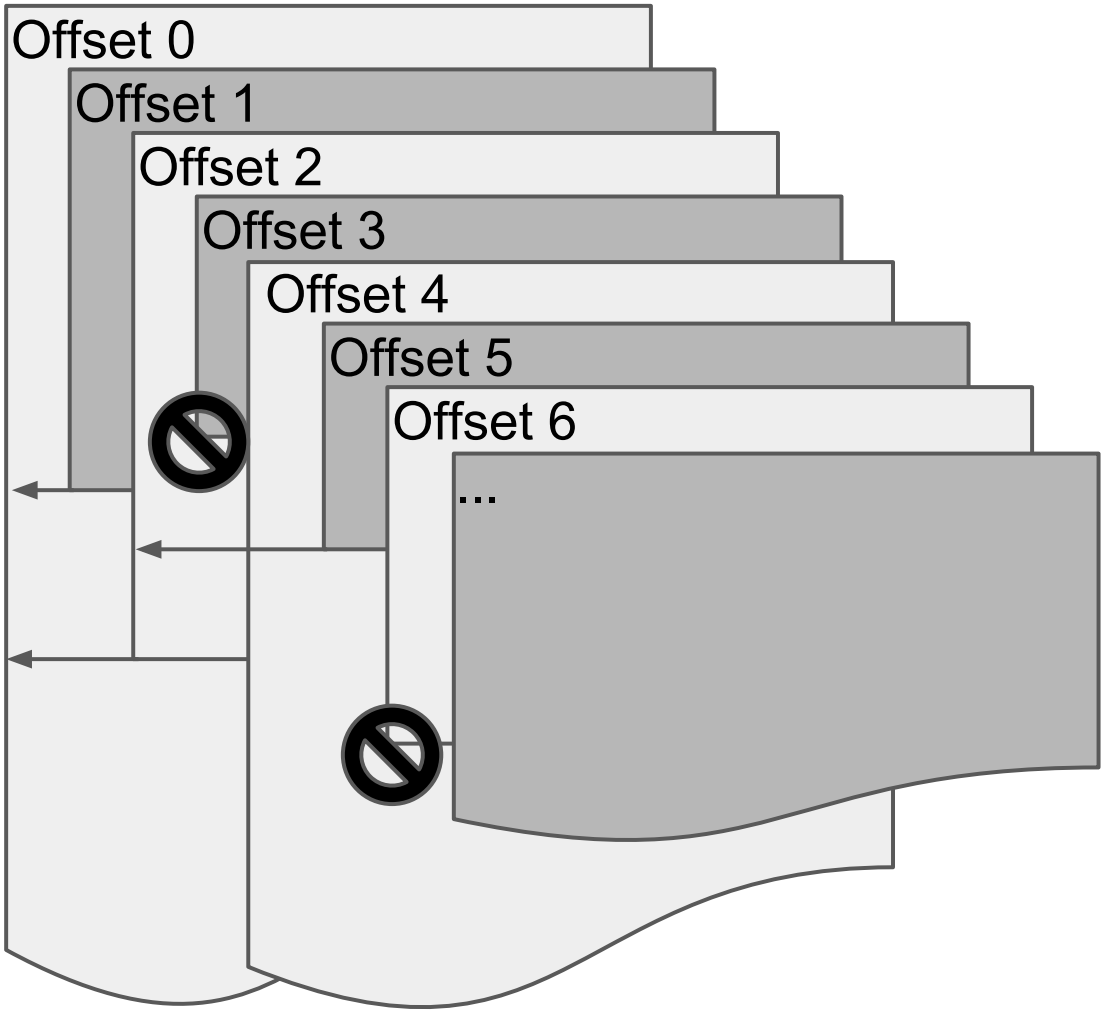
Existing Static Rewriters: w/ Heuristics

1. Assume certain compiler generated binaries
2. Assume having debug symbols
3. Assume knowledge of APIs (call backs)
4. Assume no code and data interleaving
5. Rely on relocation metadata
6. Use heuristics to recognize static memory addresses
7. ...

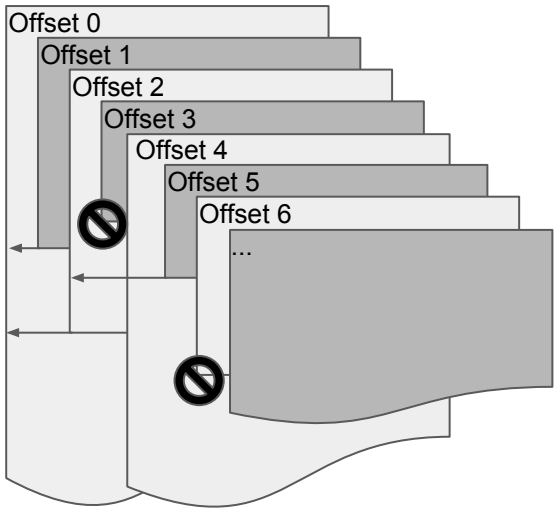
Brute Force Disassembler

"When in doubt, use brute force." – Ken Thompson

Brute Force Disassembler

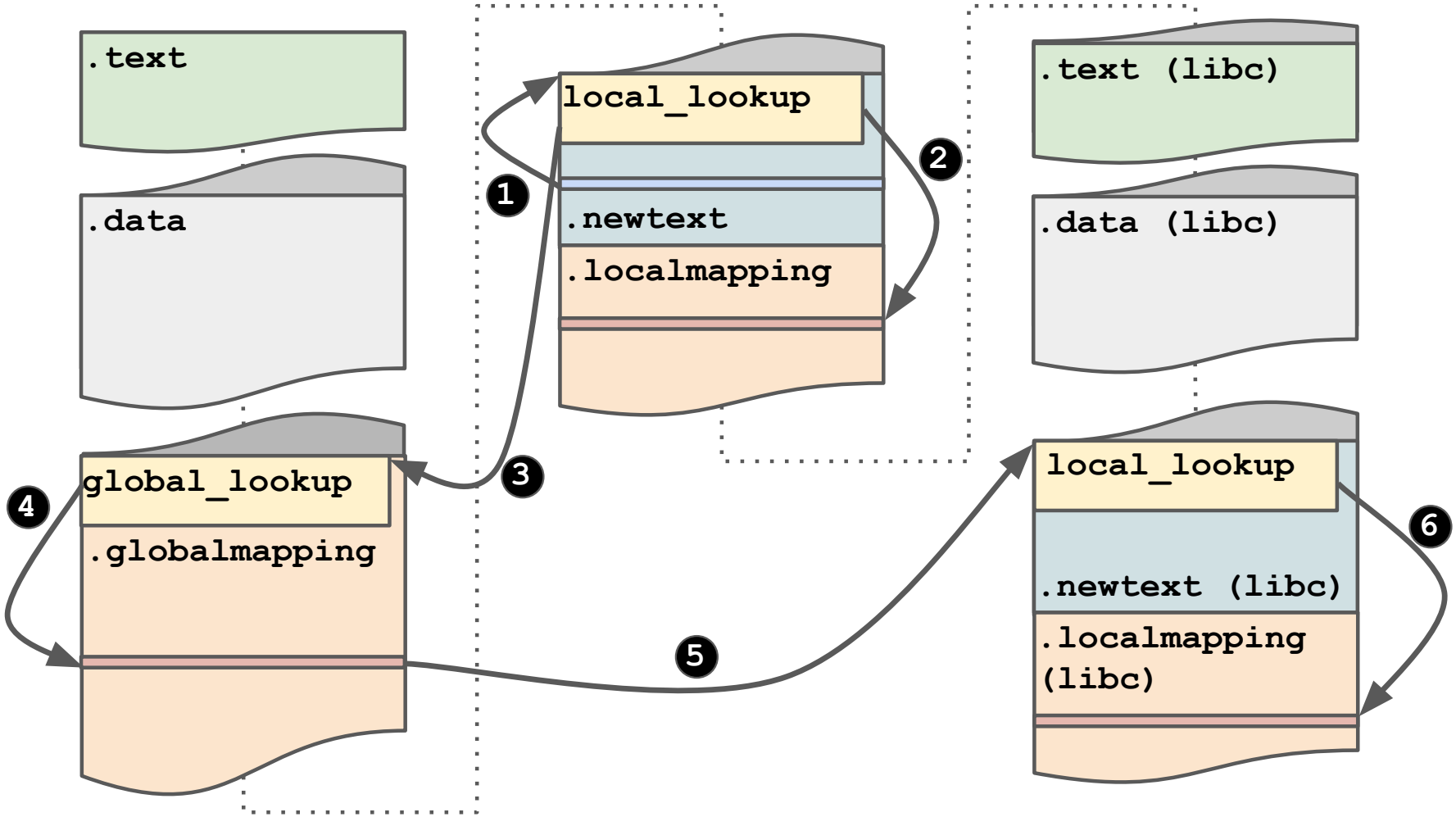


Brute Force Disassembler

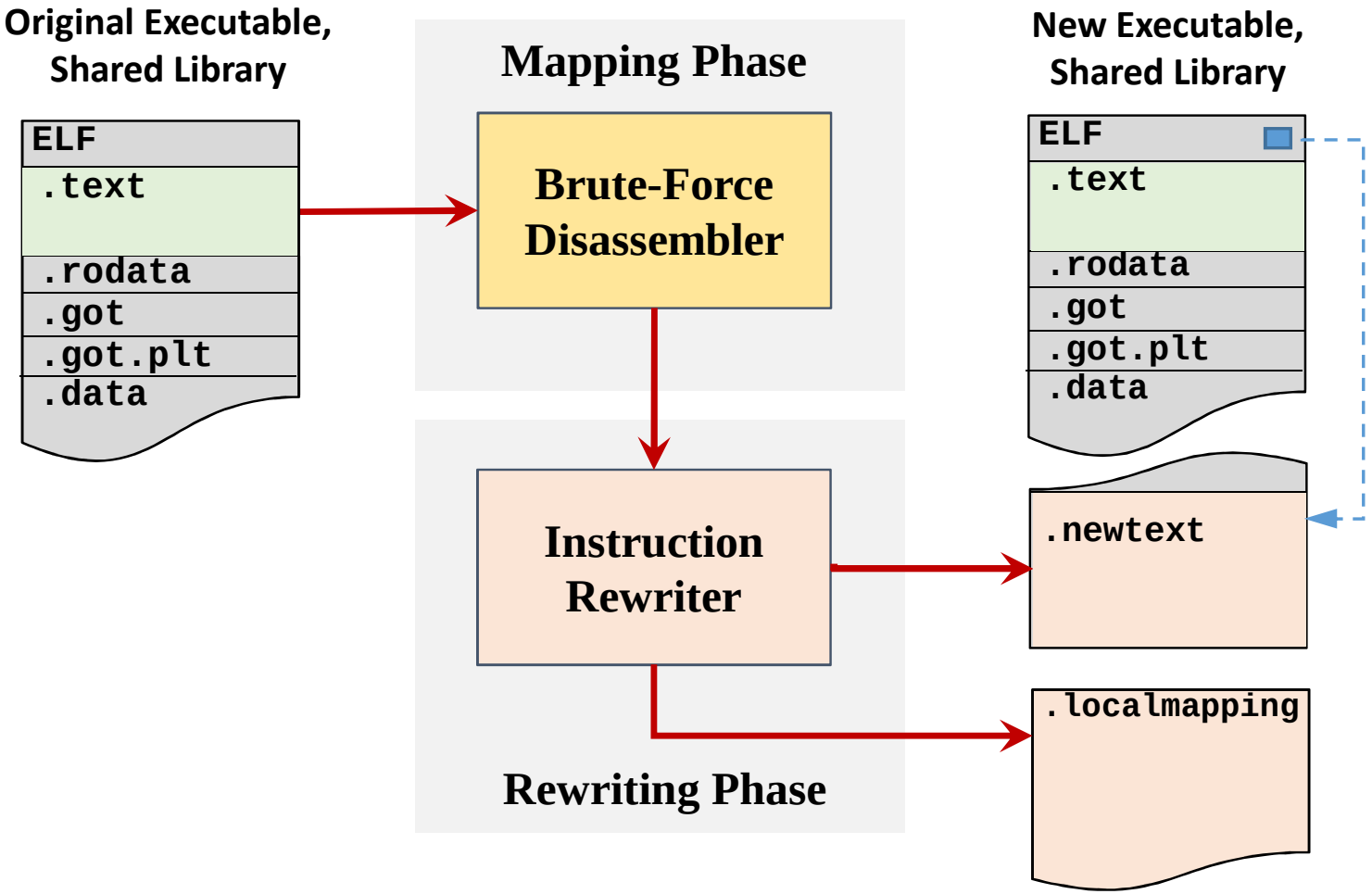


1. Statically Disassembly of Obfuscated Binaries [KRVV04]
2. Shingled Graph Disassembly [WZHK14]
3. GPU-Disasm: GPU-based x86 Disassembly [LVP+15]

Instruction Address Mapping



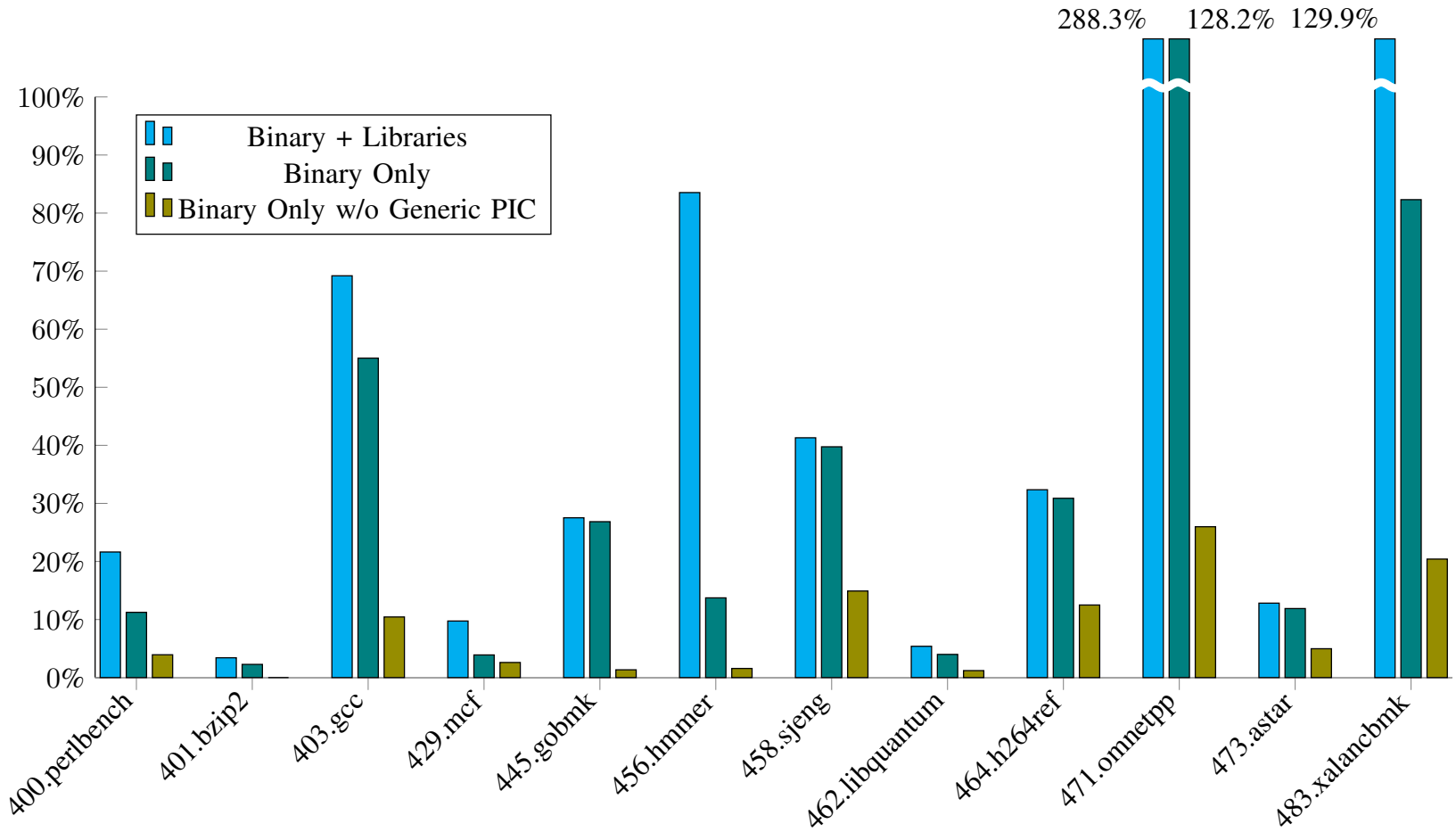
Overview of BinForce



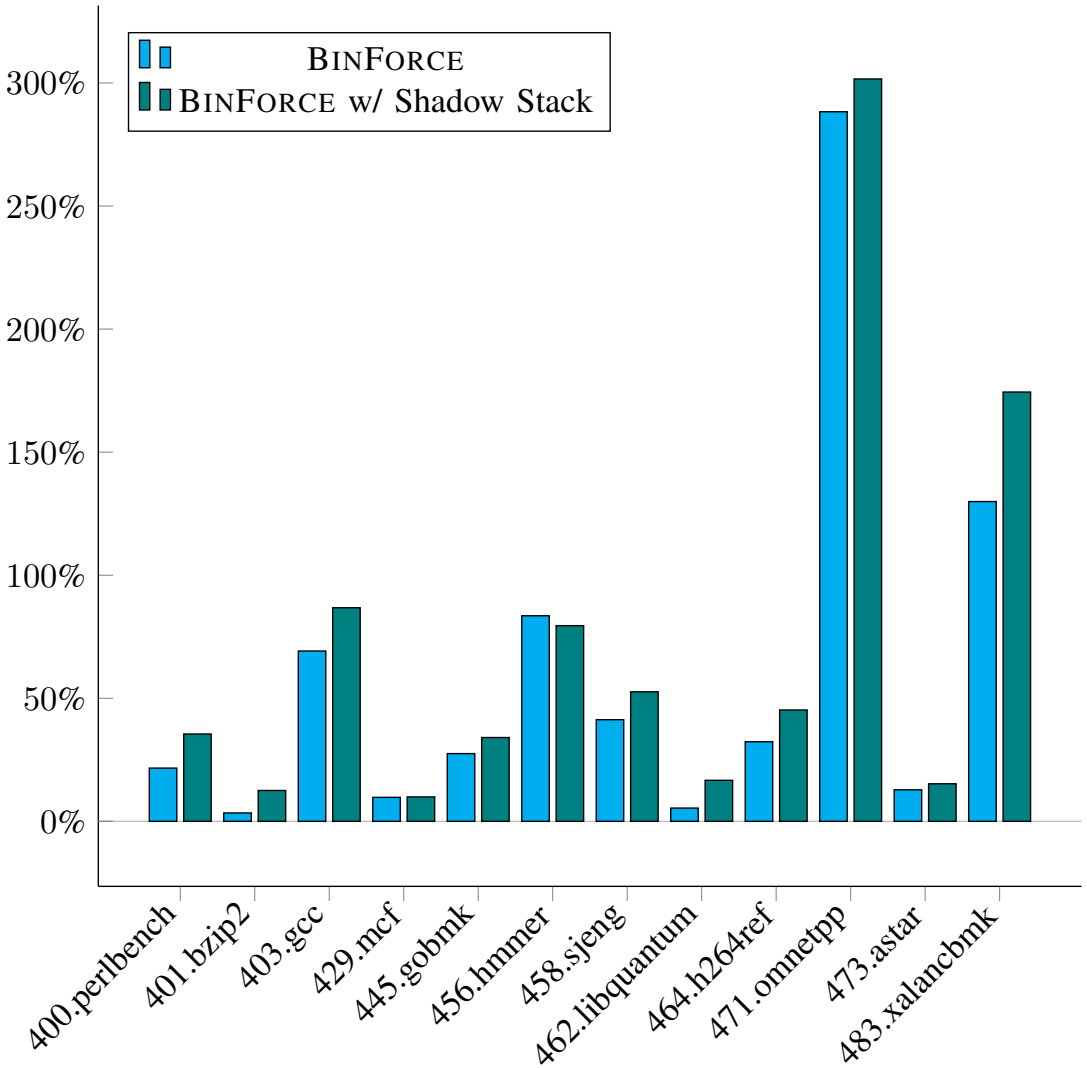
Statistics of our rewritten binaries and libraries

Benchmark	Dir. Calls	Dir. Jumps	Ind. Calls	Ind. Jumps	Cond. Jumps	Rets	.text (KB)	.newtext (KB)	Size Inc. (X)
400.perlbench	30888	24778	3896	4442	126876	22306	1047	5146	12.88
401.bzip2	1100	1050	170	152	7342	874	55	268	70.71
403.gcc	110122	64532	8916	15680	380920	45410	3225	15290	10.32
429.mcf	276	216	44	78	1300	250	12	57	202.98
445.gobmk	23548	14946	3550	3480	117378	20918	1488	6520	5.39
456.hmmer	8020	4942	556	666	28924	4106	277	1279	22.56
458.sjeng	2566	2338	256	658	12236	1570	132	604	36.17
462.libquantum	1094	758	94	146	3376	812	40	181	93.73
464.h264ref	7124	6518	1782	2000	47850	6318	520	2441	16.23
471.omnetpp	33578	10032	3830	1782	51642	14326	635	3029	13.49
473.astar	912	552	162	160	3314	750	39	184	92.52
483.xalancbmk	115154	58678	39392	14630	307122	75674	3850	17369	7.60
libc.so.6	32798	33370	9816	9012	189384	32458	1735	8435	9.77
libgcc_s.so.1	2158	2514	374	484	12862	1740	112	538	9.70
libm.so.6	5450	8870	874	892	21796	7406	277	1268	9.51
libstdc++.so.6	22456	10418	4300	4008	144516	15784	900	4258	9.53

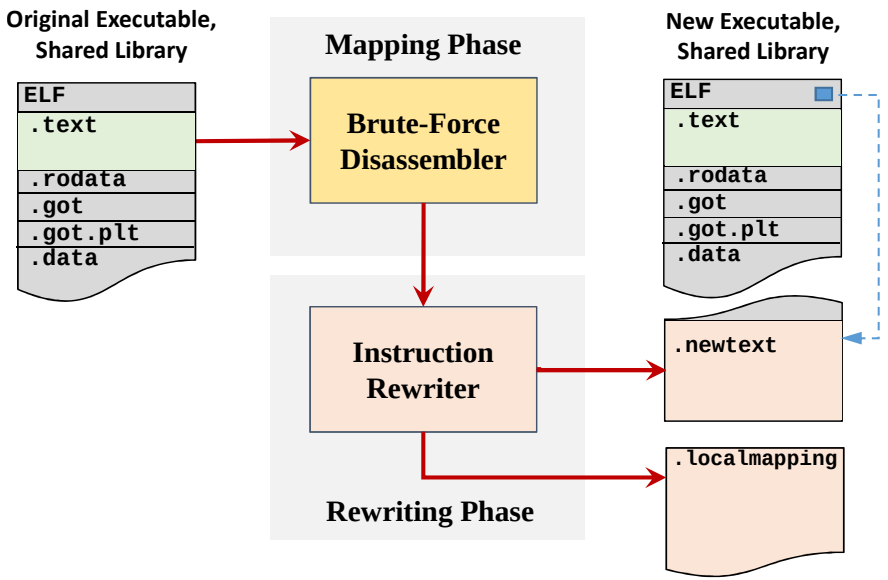
Runtime overhead for each of the benchmarks



Overhead for the benchmarks w/ shadow stack protection



Conclusion



- BinForce: **Statically rewriting** x86 binaries w/o heuristics
- Reasonable performance
- Does not support dynamically generated code
- Opportunities for optimization (e.g., size of the code, performance)

Thank You

Questions / Comments

- Erick Bauman - erick.bauman@utdallas.edu
- Zhiqiang Lin - zhiqiang.lin@utdallas.edu

Source code will be available in github.com

References I

Martín Abadi, Mihai Budiu, Úlfar Erlingsson, and Jay Ligatti, *Control-flow integrity principles, implementations, and applications*, ACM Trans. Information and System Security **13** (2009), no. 1.

Kapil Anand, Matthew Smithson, Khaled Elwazeer, Aparna Kotha, Jim Gruen, Nathan Giles, and Rajeev Barua, *A compiler-level intermediate representation based binary analysis and rewriting system*, Proceedings of the 8th ACM European Conference on Computer Systems, ACM, 2013, pp. 295–308.

Juan Caballero, Noah M. Johnson, Stephen McCamant, and Dawn Song, *Binary code extraction and interface identification for security applications*, NDSS, Feb. 2010.

Christopher Kruegel, William Robertson, Fredrik Valeur, and Giovanni Vigna, *Static disassembly of obfuscated binaries*, USENIX security Symposium, vol. 13, 2004, pp. 18–18.

Evangelos Ladakis, Giorgos Vasiliadis, Michalis Polychronakis, Sotiris Ioannidis, and Georgios Portokalidis, *Gpu-disasm: A gpu-based x86 disassembler*, International Information Security Conference, Springer, 2015, pp. 472–489.

Robert Wahbe, Steven Lucco, Thomas E. Anderson, and Susan L. Graham, *Efficient software-based fault isolation*, Proc. ACM Sym. Operating Systems Principles, 1993, pp. 203–216.

Richard Wartell, Vishwath Mohan, Kevin Hamlen, and Zhiqiang Lin, *Binary stirring: Self-randomizing instruction addresses of legacy x86 binary code*, Proceedings of the 19th ACM Conference on Computer and Communications Security (CCS'12) (Raleigh, NC), October 2012.

References II

Richard Wartell, Yan Zhou, Kevin W Hamlen, and Murat Kantarcioglu, *Shingled graph disassembly: Finding the undecideable path*, Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2014, pp. 273–285.