Z3strBV: A Solver for a Theory of Strings and Bit-vectors

Murphy Berzish[1], Sanu Subramanian[2], Yunhui Zheng[3], Omer Tripp[4], and Vijay Ganesh[1]

(1) University of Waterloo (2) Intel Security (3) IBM Research (4) Google

July 1, 2016
SMT Workshop

UNIVERSITY OF WATERLOO

Outline

- Background
- Existing solutions, motivation
- Decidability of strings+bit-vectors
- Design of Z3strBV
- Binary search heuristics
- Library-aware SMT solving
- Experimental evaluation
- Future work
- Summary and conclusion

UNIVERSITY OF WATERLOO

Background: Symbolic Execution and SMT

- Analysis of low-level programs in C/C++
- Powerful application of SMT solvers for
  - Detection of security vulnerabilities
  - Automated test case generation
- Strength of the symbolic execution engine related to expressive power, efficiency of the SMT solver backend

Why Not String-Integer Combination?

- Existing SMT solvers supporting theory of strings interpret the length of a string as an arbitrary-precision integer
- In languages like C/C++, integer values (e.g. `strlen`) are fixed-precision
- Relevant: semantics of overflow/underflow
- More efficient to model as a bit-vector and not an integer

Why Not Represent Strings as Bit-vectors?

- Strings can also be represented as (arrays of) bit-vectors
- KLEE, S2E both do this
- Performance issue: low-level bit-vector representation vs. high-level semantics of the string type
  - Path explosion: `strlen` on a symbolic string of length N forks N+1 paths.
- Difficulty in handling unbounded / arbitrary-length strings

Motivation for a String+Bit-vector Combination

- In summary, the problems with existing solutions are:
  - Strings + natural numbers has limited ability to model overflow, underflow, bit-wise operations, pointer casting, etc. without bit-vectors
  - Bit-vector solvers are not able to perform direct reasoning on strings efficiently, and cannot handle unbounded strings
- This motivates us to build Z3strBV, a solver for strings + bit-vectors.
  - Combination of a string solver (Z3str2), bit-vector solver (Z3's BV theory), bit-vector sorted length function (on top of Z3str2), and SMT solver framework (Z3)
  - Opportunity to apply new heuristics:
    - Binary search
    - Library-aware SMT solving

Contributions

- Solver for quantifier-free theory of strings, bit-vectors, and bit-vector-sorted string length
  - Built on top of the Z3str2 string solver (Zheng et al., 2015)
    - ...which is itself built on top of the Z3 SMT solver (de Moura, Bjorner, et al., 2008)
  - Extensions for bit-vector sorts, in particular $\texttt{strlen}_{bv}$`: String -> Bitvector`
- New solver heuristics:
  - Binary search pruning strategy to reach consistent length assignments
  - Library-aware SMT solving for improved performance
- Decidability of string+bit-vector combination

Motivating Example

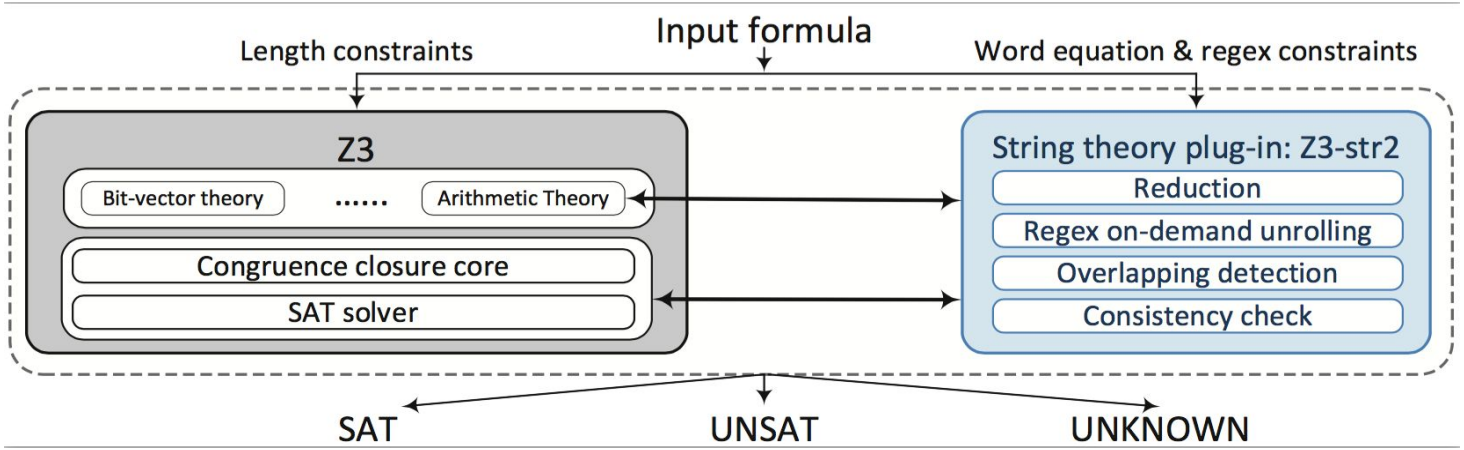
```
bool check_login(char *username, char *password) {
  if (!validate_password(password)) {
    invalid_login_attempt(); exit(-1);
  }
  const char *salt = get_salt8(username);
  uint16_t len = strlen(password) + strlen(salt) + 1;
  if (len > 32) { invalid_login_attempt(); exit(-1); }
  char *saltedpw = (char*)malloc(len);
  strcpy(saltedpw, password);
  strcpy(saltedpw, salt);
  ...
}
```

Decidability of String + Bit-Vector Combination

- The satisfiability problem for the QF theory of word equations, bit-vector length, and bit-vector terms is decidable.
- Proof sketch: by reduction to strings + regular language membership
  - Shown decidable by Schulz (1992)
- This may seem trivial -- finitely many BVs implies finitely many strings?
  - **NO!** Overflow semantics apply to length terms too
- Decidability is in fact non-trivial as infinitely many strings must be considered

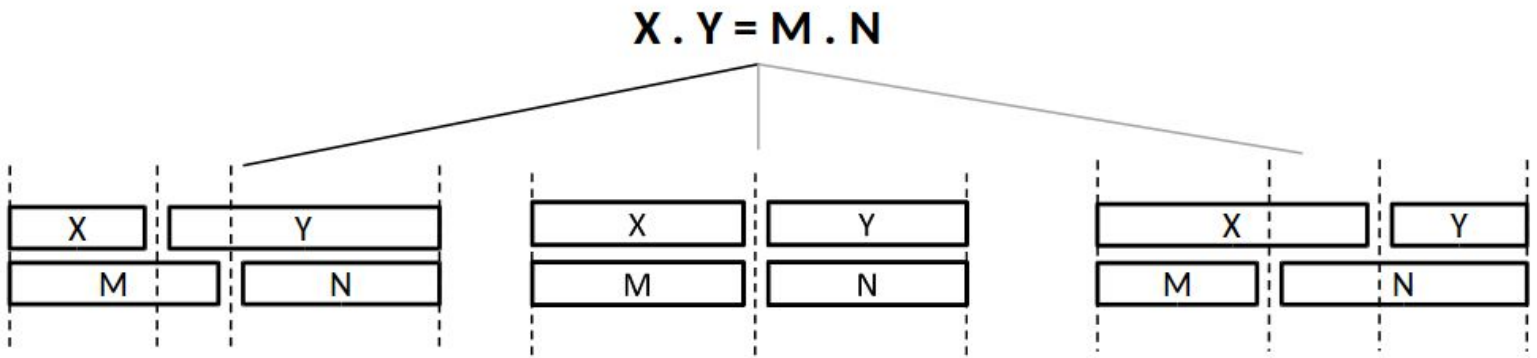
Design Overview

- Word equation solving
- Integration of string and bit-vector theory
- Binary search heuristic for search-space pruning



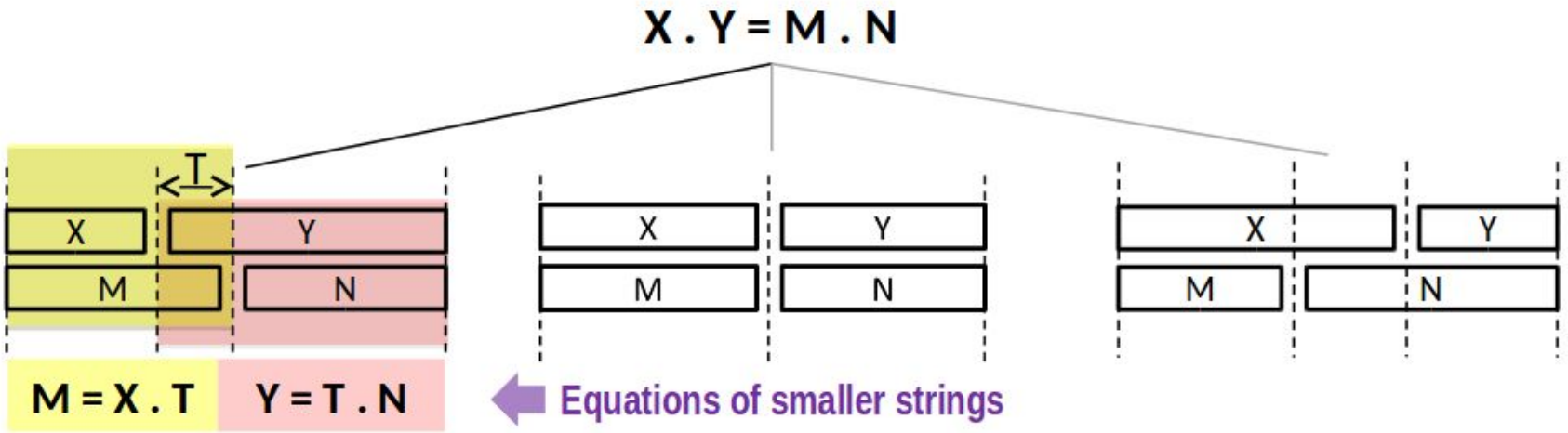
String Equation Solving

- Key technique of Z3str2: recursively split equations into subproblems until the system can be solved directly
- Given an equation, identify all possible splits / “arrangements”



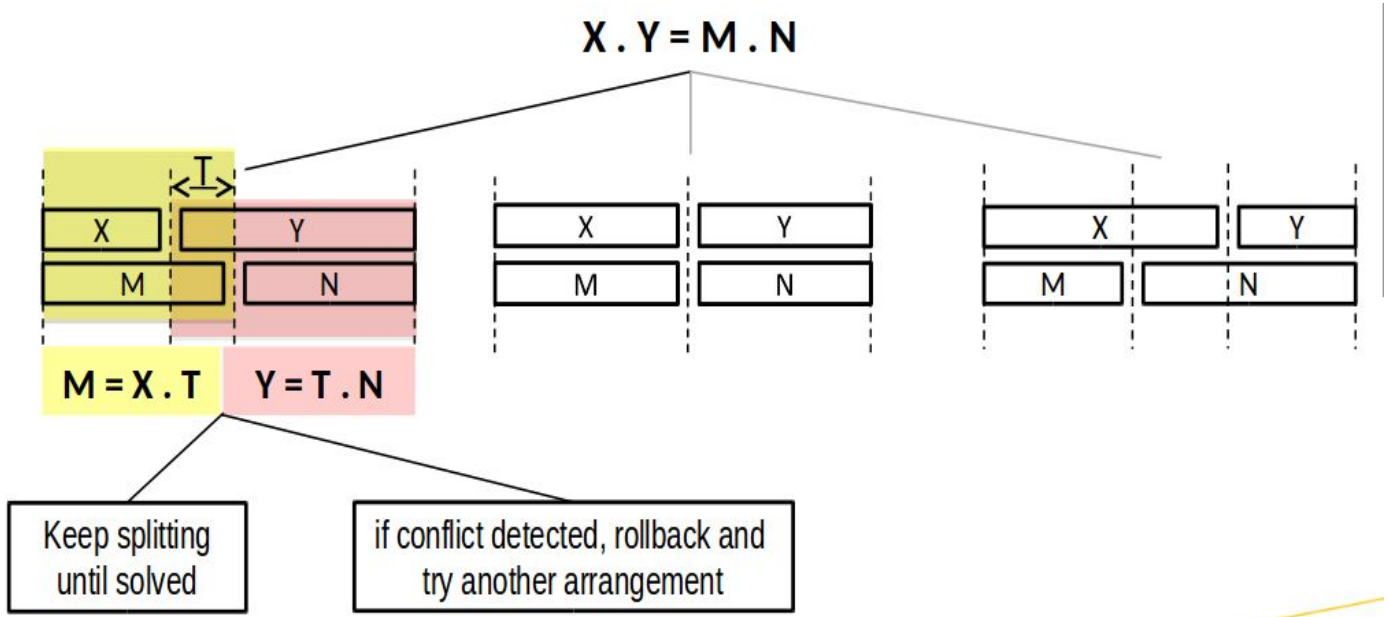
String Equation Solving

- Given an arrangement, generate a set of sub-equations over smaller strings



String Equation Solving

- Given an arrangement, generate a set of sub-equations over smaller strings
- New equations are split recursively until all equations are between variables and string constants



String-Bitvector Theory Integration

- Three main rules:
  - Each character has length 1, the empty string has length 0
  - $X = Y \Rightarrow \text{strlen}_{bv}(X) = \text{strlen}_{bv}(Y)$
  - $W = X \,.\, Y \,.\, Z \ldots \Rightarrow \text{strlen}_{bv}(W) = \text{strlen}_{bv}(X) + \text{strlen}_{bv}(Y) + \text{strlen}_{bv}(Z) + \ldots$
- These are, elegantly, of similar form to the rules for string-integer integration
- Overflow semantics handled by bit-vector theory solver

Binary Search Heuristic

- Z3str2 performs (naive) linear search for the length of variables
  - Constraints of the form “len(X) > 15000” are checked starting at “len(X) = 0, 1, 2, 3, …”
- Z3strBV performs binary search over bit-vector lengths
  - e.g. searching for a 2-bit length L: midpoint is 2, branch on len(X) < 2, len(X) = 2, len(X) > 2
  - If strings are longer than the upper bound, overflow semantics come into play
  - Consistent lengths found in significantly less time
  - This is sound and very efficient
- Similar technique back-ported to the integer version
  - Main difference: no *a priori* fixed upper bound for integers
  - Choose a “floating” upper bound that the solver can choose to increase if necessary

Library-Aware SMT Solving

- Provide native solver support for library functions that are:
  - Available in popular programming languages like C/C++
  - Very commonly used by programmers
  - A frequent source of errors due to programmer mistakes
  - Expensive to analyze symbolically due to large number of potential paths
- Extend the logic of traditional SMT solvers with declarative summaries of functions such as `strlen`, `strcpy`, etc.
- Preliminary work with Z3strBV to support these functions

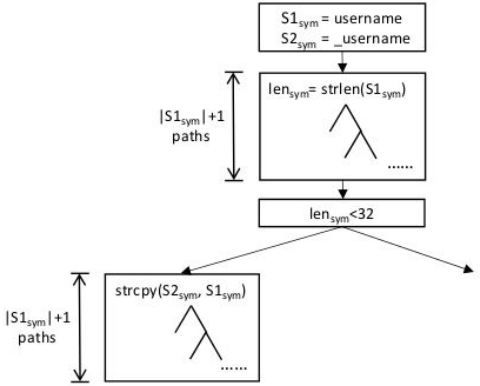
Experimental Results

- We evaluated our solver on 7 real buffer overflow vulnerabilities:
  - CVE-2015-3824: Google stagefright ’tx3g’ MP4 atom integer overflow
  - CVE-2015-3826: Google stagefright 3GPP metadata buffer overread
  - CVE-2009-0585: libsoup integer overflow
  - CVE-2009-2463: Mozilla Firefox/Thunderbird Base64 integer overflow
  - CVE-2002-0639: Integer and heap overflows in OpenSSH 3.3
  - CVE-2005-0180: Linux kernel SCSI IOCTL integer overflow
  - FreeBSD wpa supplicant(8) Base64 integer overflow
- Handcrafted constraints for vulnerable region
- String+bit-vector **generated a model** for all instances
- String+integer **could not solve** any instances

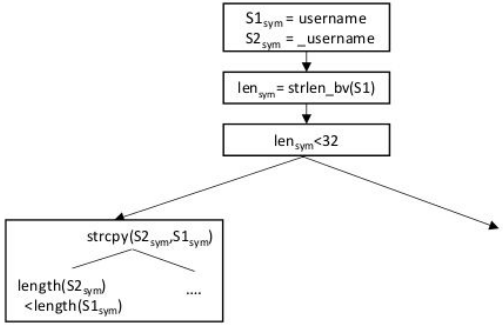
Vulnerability	Z3strBV	Z3str2
CVE-2015-3824	0.079s	TO (1h)
CVE-2015-3826	0.108s	TO (1h)
CVE-2009-0585	0.031s	TO (1h)
CVE-2009-2463	0.279s	TO (1h)
CVE-2002-0639	0.116s	TO (1h)
CVE-2005-0180	0.029s	TO (1h)
FreeBSD Bugzilla #137484	0.038s	TO (1h)

Experimental Results

- Evaluation of library-aware SMT solving via comparison with KLEE
- Input constraints from the motivating example (`check_login`)
- The size of the length variable determines the total number of paths
- We consider 8-bit and 16-bit length variables
  - KLEE times out after 120 minutes with a 16-bit length
  - Z3strBV finds the bug in 0.27 seconds
- The path constraints are not hard; there are just too many paths



(a) KLEE



(b) Library-aware Solving

Experimental Results

- Binary search heuristic applied to unconstrained string variables
- Implemented a modified Z3strBV that uses linear search
- Significant gain in performance when binary search is used

Z3strBV	SAT				UNSAT				TIMEOUT (20s)				UNKNOWN				Total	
	#	T_{min}	T_{avg}	T_{max}	#	T_{min}	T_{avg}	T_{max}	#	T_{min}	T_{avg}	T_{max}	#	T_{min}	T_{avg}	T_{max}	#	Time(s)
Binary Search	98	0.060	0.172	2.667	9	0.047	0.081	0.320	0	0	0	0	2	0.051	0.085	0.118	109	17.7 (**1x**)
Linear Search	72	0.060	0.097	0.618	9	0.061	0.111	0.415	27	20.000	20.000	20.000	1	0.072	0.072	0.072	109	548.1 (**31x**)

Experimental Results

- Performance of binary search heuristic in the integer version (Z3str2)
- Compared against the previous (linear search) Z3str2, and CVC4
- Z3str2 with binary search is faster than both linear-search Z3str2 and CVC4

	Z3-str2 (Binary Search)	**Z3-str2 (Naive Search)**	**CVC4**
SAT	169	138	126
UNSAT	34	34	19
UNKNOWN	2	2	0
Timeout	0	31	60
Tool reports error	0	0	0
Crash	0	0	0
Total no. of benchmarks	205	205	205
Total time (sec)	41.697 (**1x**)	9569.639(**229x**)	12014.893(**264x**)

UNIVERSITY OF WATERLOO

Future Work

- Tighter integration with symbolic execution engines
  - String + bit-vector in KLEE, S2E
  - String + integer into Jalangi
- Development of efficient function summaries for string functions in the standard libraries of several programming languages
- Integration of Z3str2 and Z3strBV into the main Z3 codebase
  - ***The port to the newest version of Z3 is now feature-complete and in testing.***

Summary and Conclusion

- Motivation and design for a solver for strings + bit-vectors
  - String+integer less efficient than string+bit-vector for overflow/underflow
  - Bit-vector solvers are inefficient at modelling strings as arrays of bit-vectors
- Binary search heuristic for consistent length assignments
  - Useful for both bit-vector and integer length terms
  - Significant performance improvements vs. state-of-the-art solvers
- Library-aware SMT solving
  - Large performance improvements over traditional symbolic execution techniques

References

- Yunhui Zheng, Xiangyu Zhang, and Vijay Ganesh. Z3-str: A Z3-based string solver for web application analysis. In Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2013, pages 114–124, 2013.
- Yunhui Zheng, Vijay Ganesh, Sanu Subramanian, Omer Tripp, Julian Dolby, and Xiangyu Zhang. Effective search-space pruning for solvers of string equations, regular expressions and length constraints. In Daniel Kroening and Corina S. Pasareanu, editors, Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part I, volume 9206 of Lecture Notes in Computer Science, pages 235–254. Springer, 2015.
- “A case study of concolic testing tools and their limitations.“, Qu, Xiao, and Brian Robinson. Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium on. IEEE, 2011.
- Vijay Ganesh and David L. Dill. A decision procedure for bit-vectors and arrays.In Proceedings of the 19th International Conference on Computer Aided Verification, CAV’07, pages 519–531, 2007.
- Leonardo De Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In Proceedings of the Theory and practice of software, 14th international conference on Tools and algorithms for the construction and analysis of systems, TACAS’08, pages 337–340, 2008.