Generating Vulnerability Signatures for String Manipulating Programs Using Automata-based Forward and Backward Symbolic Analyses

Fang Yu, Muath Alkhalaf, Tevfik Bultan {yuf, muath, bultan}@cs.ucsb.edu

Overview

- We present automata-based symbolic string analyses for automatic verification of string manipulating programs
- We compute the *pre*- and *post*-conditions of common string functions using deterministic finite automata (DFAs)
- We compute DFAs that characterize *all* possible values that string expressions can take in any possible execution of a program using forward and backward symbolic analyses

Why do we need string analysis?

The top three vulnerabilities in OWASPs top ten list (which lists the most serious web application vulnerabilities) are due to improper manipulation of strings:

1. Cross Site Scripting (XSS)
2. Injection Flaws (such as SQL injection)
3. Malicious File Execution (MFE)

What does our string analysis achieve?

- Detects vulnerabilities in web applications that are due to string manipulation
- Proves the absence of vulnerabilities in web applications that use proper sanitization
- Generates a characterization of all malicious inputs that may compromise a vulnerable web application

An example

```
1 <?php
2    $www = $_GET["www"];
3    $l_otherinfo = "URL";
4    $www = preg_replace(
        "/[^A-Za-z0-9 .-@://]/",
        "",
        $www);
5    echo $l_otherinfo . ": " . $www ;
6 ?>
```

- The `echo` statement in line 5 can contain a Cross Site Scripting (XSS) vulnerability
- A malicious user may provide an input that contains the string constant `<script` and execute a command leading to a XSS attack
- The goal of the replace statement in line 4 is to remove any special characters from the input to prevent such attacks

Vulnerability

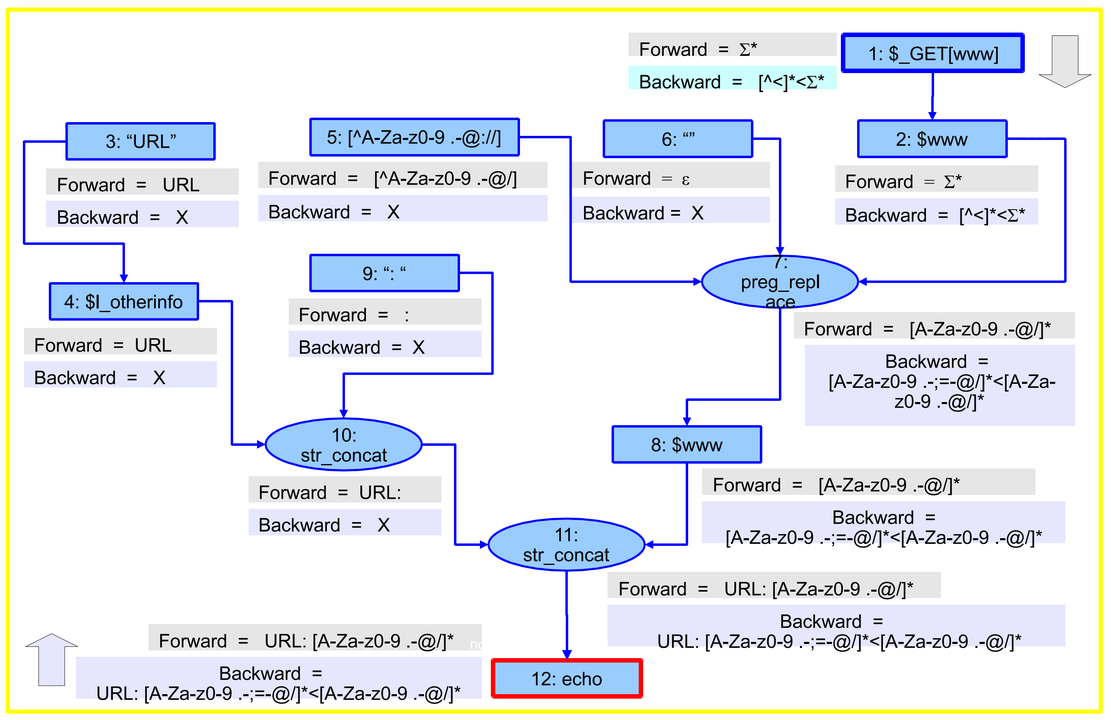
- The replace operation in line 4 contains an error that leads to a XSS vulnerability
- The error is in the match pattern of the replace operation: `[^A-Za-z0-9 .-@://]`
- The expression `.-@` should have been `.\-@`

Detecting vulnerabilities

- We detect vulnerabilities using forward analysis
- Use automata-based forward symbolic analysis to identify the possible values of each string expression (uses post-condition computation)
- Intersect the result of the *sink* nodes with the attack pattern
- If the intersection is empty then the program is not vulnerable with respect to the attack pattern. Otherwise, it is vulnerable
- We use *language*-based replacement [4] to model `preg_replace()` and PHP sanitization routines, e.g. `addslashes()` and `mysql_real_escape_string()`

Generating vulnerability signatures

- We generate vulnerability signatures using backward analysis
- A vulnerability signature is a characterization that includes all malicious inputs that can be used to generate attack strings
- Use backward analysis starting from the sink nodes and traverse the dependency graph backwards to find out malicious inputs (uses pre-condition computation)
- Both forward and backward analyses use an automata-based widening operator [1] to accelerate fixpoint computations

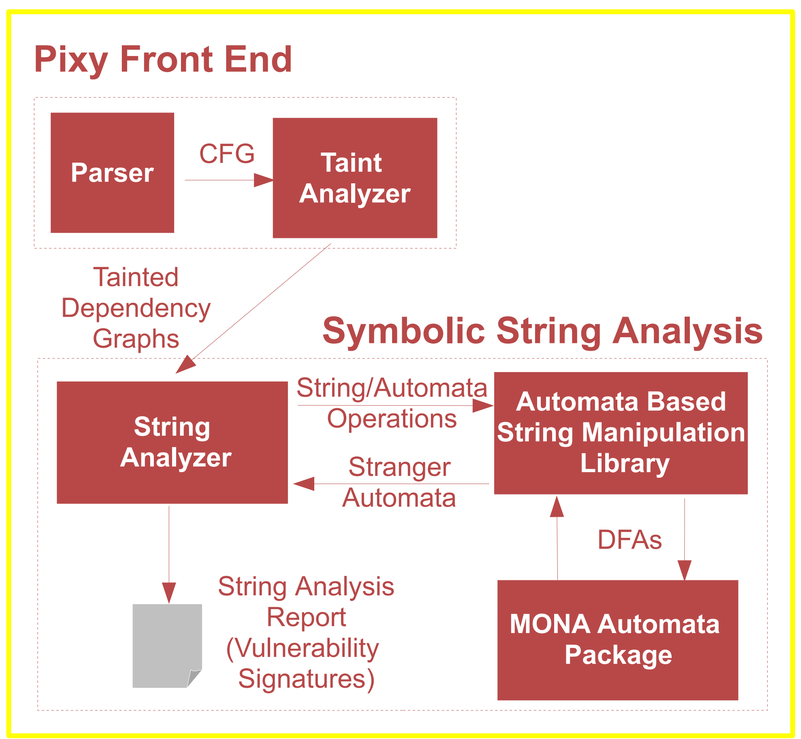


The Result of Forward and Backward Analyses

Implementation

We use Pixy [3] as a front end and MONA [2] automata package for automata manipulation. The implementation consists of the following parts:

- *PHP Parser*: Parses the PHP program and constructs the control flow graph (CFG)
- *Taint Analyzer*: Identifies *sinks* (sensitive functions that may use tainted data) and generates their dependency graphs using alias and dependency analyses. If no sinks are found, the application is not vulnerable
- *String Analyzer*: Implements vulnerability (forward and backward) analysis on dependency graphs for all sinks that are found
- *String Manipulation Library (SML)*: Handles all core string and automata operations such as concatenation, prefix, suffix, replace, intersection, union, and widen



Contributions

- Sound verification techniques for PHP web application vulnerability analysis and vulnerability signature generation, focusing on SQLCI, XSS and MFE attacks
- Combining forward and backward symbolic string analyses for vulnerability signature generation
- Implementation of forward and backward image computations for string operations (including complex operations such as `preg_replace()`) using a symbolic automata representation (MBDDs)
- The first automata-based string analysis tool that can automatically generate vulnerability signatures of vulnerable PHP programs
- The implementation and benchmarks are available at: `http://www.cs.ucsb.edu/~vlab/stranger`

UCSB

Verification Laboratory (VLab)
Department of Computer Science
University of California, Santa Barbara

References

[1] Constantinos Bartzis and Tevfik Bultan. Widening arithmetic automata. In *Proceedings of the 16th International Conference on Computer Aided Verification*, pages 321–333, 2004.

[2] BRICS. The MONA project. `http://www.brics.dk/mona/`.

[3] Nenad Jovanovic, Christopher Krügel, and Engin Kirda. Pixy: A static analysis tool for detecting web application vulnerabilities (short paper). In *Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P 2006)*, pages 258–263, 2006.

[4] Fang Yu, Tevfik Bultan, Marco Cova, and Oscar H. Ibarra. Symbolic string verification: An automata-based approach. In *15th International SPIN Workshop on Model Checking Software (SPIN 2008)*, pages 306–324, 2008.