Finding SQL Injection Vulnerabilities in Server-side SQL Libraries using Symbolic Execution

Kangqi Ni Xiangyu Li Taesoo Kim
Georgia Institute of Technology

Motivation

- SQL injection attacks used to be a severe security threat
- Nowadays mitigated by the use of server-side SQL libraries to pre-process user input before submitting to the database.
  - Sensitive characters such as quotation marks are escaped.

Motivation

- Inadequate validation of the user data in the SQL libraries would cause serious security issues
  - Users of the libraries typically trust the library code
  - Potential large population of certain SQL libraries

Overview

- We propose an automated technique that tries to find SQL injection vulnerabilities in server-side SQL libraries.
  - Execute library code symbolically.
  - Examine the relation between the user input and the SQL statement passed to the database.
  - There is a potential vulnerability if the pre-processed user input still contains sensitive characters.

Past Approach

- Defensive Coding
- Dynamic Monitoring
- Black-box Test Generation
- Taint Analysis
- Query Inspection
  - e.g., sql library

Our Problem Scope

- Pick sql library as subject, and verify its soundness

Symbolic Execution

- Reason about program behaviors on potentially infinite set of possible input

- Produce a concrete input / trace that leads execution to reach a particular program point

Our Approach

1. Automatically generate inputs with the help of concolic execution
2. Dynamically track taint to determine how input affects sensitive sinks
3. Mutate inputs to produce exploits by replacing tainted part with shady strings [1]

[1] HAMPI: A Solver for String Constraints, *ISSTA 2009*

Evaluation

We'll run our technique on selected server-side SQL libraries to discover injection vulnerabilities.

This project is risky by nature.

- There might be no injection vulnerabilities in our subjects.
- Exploration space is limited by the symbolic execution library and our computational resources. The exploration space may not be sufficient to discover vulnerabilities.
- The ability of solving the constraint is limited by the constraint solver. Our proposed technique involved solving complicated string-related constraints, which is known to be difficult.

We'll also use manually crafted simple subjects with known vulnerabilities.

Plan

-- 11/3/2014. Explore past research on related topics

-- 11/17/2014. Implement the automated symbolic exploration of the SQL libraries. Implement constraint solving (and probably, some optimizations).

-- 11/24/2014. Conduct experiments on selected SQL libraries to see whether we could discover injection vulnerabilities.

-- 12/1/2014. Prepare project demo and presentation. Artifacts are camera-ready.