**Contens**

1 Web Security……………………………………………………………………………………………………….2

1.1 Main contributions .................................................................................................2

1.2 Running Example ....................................................................................................2

1.2.1 Information ..................................................................................................2

1.2.2 Example .......................................................................................................3

1.3 Background ...........................................................................................................4

1.4 The CNF formula .....................................................................................................4

Web Security

## **Main contributions**

1. We define SmpSQL, an SQL fragment which is contained in FO2bd;
2. We define a a simple imperative script language SmpSL with embedded SmpSQL state-ments;
3. We give a construction for weakest preconditions in FO2bd for SmpSL;
4. We implemented the weakest precondition computation for SmpSL;
5. We implemented a decision procedure for FO2bd. The procedure is based on the decidability and NEXPTIME completeness result for FO2 by [21], but we use a more involved algorithm which reduces the problem to a SAT solver and is optimized for performance.

We evaluate our methodology on three applications: a web administrator, a simple firewall, and a conference management system. We compared our tool with Z3 [14], currently the most advanced general-purpose SMT solver with (limited) support for quantifiers. In general, our tool performs better than Z3 in several examples for checking the validity of verification conditions of SmpSL programs. However, our tool and Z3 have complementary advantages: Z3 does well for unsatisfiable instances while our tool performs better on satisfiable instances. We performed large experiments with custom-made blown up versions of the web administrator and the firewall examples, which suggest that our tool scales well. Moreover, we tested the scalability of our approach by comparing of our underlying FO2 solver with three solvers on a set of benchmarks we assembled inspired by combinatorial problems. The solvers we tested against are Z3, the SMT solver CVC4 [3], and the model checker Nitpick [7]. Our solver outperformed each of these solvers on some of the benchmarks.

## **Running Example**

### Information

We introduce our approach on the example of a simple web service.The example is a translation from PHP with embedded SQL commands into SmpSL of code excerpts from the Panda web-administrator. The web service provides several services implemented in dedicated functions for subscribing a user to a newsletter,

deleting a newsletter, making a user an admin of a newsletter, sending emails to all subscribed users of a newsletter, etc. We illustrate our verification methodology by exposing an error in the Panda web-administrator. The verification methodology we envision in this paper consists of (1) maintaining database invariants and (2) verifying a contract specification for each function of the web service.

The database contains several tables including NS =NewsletterSubscription with at-butes nwl,user,subscrided and code. The database is a structure whose universe is partitioned into three sets: domU, boolB, and codesB. The attributes nwl and user range over the finite set domU, the attribute subscribed ranges over boolB = {True,False} and the attribute code ranges over the fixed finite set boolB. The superscripts in domU, boolB, and codesB serve to indicate that the domain domU is unbounded, while the Boolean domain and the domain of codes are bounded (i.e. of fixed finite size). When s=true,(n,u,s,c )∈NS signifies that the user u is subscribed to the newsletter n. The process of being (un)subscribed from/to a newsletter requires an intermediary confirmation step in which the confirm code c plays a role.

### Example

Provides the functions subscribe, unsubscribe, and confirm translated manually into SmpSL. 1 The comments in quotations // “. . .” originate from the PHP source code. The intended use of these functions is as follows: In order to subscribe a user u to a newsletter n, the function subscribe is called with inputs n and u (for example by a web interface operated by the newsletter admin or by the user). subscribe stores the tuple(n,u,false,new\_code) in NS, where new\_code is a confirmation code which does not occur in the database, and an email containing a confirmation URL is sent to the user u. Visiting the URL triggers a call to confirm with input new\_code, which subscribes u to n by replacing the tuple(n,u,false,new\_code) of NS to with (n,u,true,nil). For unsubscribe the process is similar, and crucially, unsubscribe uses the same confirm function. confirm decides whether to subscribe or unsubscribe according to whether n is currently subscribed to u. The CHOOSE command selects one row non-deterministically.

The database preserves the invariant

|  |  |
| --- | --- |
| . | (1) |

Inv says that the pair (n, u) of newsletter and user is a key of the relation NS. The subscripts of the quantifiers denote the domains over which the quantified variables range. In our verification methodology we add invariants as additional conjuncts to the pre- and postconditions of every function. In this way invariants strengthen the pre-conditions and can be used to prove the post-conditions of the functions. On the other hand, the post-conditions require to re-establish the validity of the invariants.

Provides pre- and post-conditions pref and postf for each of the three functions f. The relation names d, b, and c are interpreted as the sets domU, boolB, and codesB, respectively. Proving correctness amounts to proving the correctness of each of the Hoare triples {〖pre〗\_f ∧Inv} **f** {〖post〗\_f ∧Inv}.

## **Background**

|  |  |
| --- | --- |
| , | (2) |

where Precision - defined in Equation 1

tp - the number of true positives identified by the model;

fp - the number of false positives identified by the model.

## **The CNF formula**

* The universe As is As;
* An unary relation name U is interpreted as the set e1;
* A binary relation name R is interpreted as the set e2.

**Table 1** – Genetic algorithm parameters for simple login script

|  |  |
| --- | --- |
| Parameter | Values |
| Population Size | 30 |
| Survivor | 3 |
| Maximum # Generation | 20 |
| # input within one individual | 2 |
| Type of inputs | Strings |
| Crossover rate (Probability) | 0.5 |
| Mutation rate (Probability) | 0.5 |



**Figure 1** – PHP Script of Newspaper Display Script