Project: DRA FRONT END FILTER PROJECT

Title: Security Conjecture for the SSQL Abstract Machine

Ref: DS/FMU/FEF/006

Issue: Revision: 4.2

Date: 5 June 2016

Status: Approved

Type: Specification

Keywords:

Author:

Name: G. M. Prout
Location: WIN01

Authorisation for Issue:

Name: R.B. Jones
Function: HAT Manager

Abstract: The formal specification of the SSQL abstract machine and the conjecture to be proven in order to prove its security. This contributes to the DRA front end filter project RSRE 1C/6130.

Distribution: HAT FEF File
Simon Wiseman
0 DOCUMENT CONTROL

0.1 Contents List

<table>
<thead>
<tr>
<th>DOCUMENT CONTROL</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1 Contents List</td>
<td>2</td>
</tr>
<tr>
<td>0.2 Document Cross References</td>
<td>2</td>
</tr>
<tr>
<td>0.3 Changes History</td>
<td>3</td>
</tr>
<tr>
<td>0.4 Changes Forecast</td>
<td>3</td>
</tr>
</tbody>
</table>

1 GENERAL

| 1.1 Scope | 4 |
| 1.2 Introduction | 4 |

2 PRELIMINARIES

3 CONSTRUCTION OF AN SSQL ABSTRACT MACHINE

| 3.1 Types of Components | 4 |
| 3.2 Building a Transition Function | 5 |
| 3.3 The SSQL Abstract Machine | 5 |

4 SECURITY CONJECTURE

| 4.1 Behavioural Abstraction | 6 |
| 4.2 SSQL Abstract Machine Correctness Conjecture | 7 |

5 CLOSING DOWN

6 THE THEORY fef006

| 6.1 Parents | 8 |
| 6.2 Children | 8 |
| 6.3 Constants | 8 |
| 6.4 Type Abbreviations | 8 |
| 6.5 Definitions | 8 |

7 INDEX

<table>
<thead>
<tr>
<th>0.2 Document Cross References</th>
<th>10</th>
</tr>
</thead>
</table>
0.3 Changes History


Issue 4.3 Removed dependency on ICL logo font

0.4 Changes Forecast

None.
1 GENERAL

1.1 Scope

This document gives a formal specification of the SSQL abstract machine and a formal specification of the security conjecture required to be proven in order to prove the security of the SSQL semantics. It constitutes deliverable D4 of work package 1a, as given in section 7 of the Secure Database Technical Proposal, [1].

1.2 Introduction

In the Secure Database Technical Proposal, [1], we stated that we propose to formalise the semantics of SSQL in such a way as to separate out security considerations from other aspects of the semantics. In this document, we describe how to construct an SSQL abstract machine from a transition function and an initial state, \( \text{isstate} \). The transition function is built from the three components \( \text{hide} \), \( \text{processQuery} \) and \( \text{updateState} \). The function \( \text{hide} \), defined in [3], operates on the state of the database and returns a ‘sanitised’ version of the state. The function \( \text{processQuery} \), defined in [4], computes the result of a query on the ‘sanitised’ state of the database. The function \( \text{updateState} \), defined in [3], checks the security of the result of the query processing and only applies the update to the database if it is secure.

2 PRELIMINARIES

The following ProofPower instructions set up the new theory \( fef006 \) and set the context for the proof tools.

```sml
open theory "fef014";
(force_delete_theory "fef006" handle _ => ());
new_theory "fef006";
push_pc "hol1";
```

3 CONSTRUCTION OF AN SSQL ABSTRACT MACHINE

3.1 Types of Components

As stated in [1], we model an SSQL abstract machine as a pair consisting of a state transition function and an initial state. The transition function is to be built from the three components \( \text{hide} \) and \( \text{updateState} \), specified in [3], and \( \text{processQuery} \), specified in [4]. We first give abbreviation definitions for the types \( \text{Hide} \) of the \( \text{hide} \) component, \( \text{Process} \) of the \( \text{processQuery} \) component and \( \text{Ustate} \) of the \( \text{updateState} \) component.

```sml
declare_type_abbrev("Hide",Class × State → State");
```
SML

```sml
declare_type_abbrev("Process", [], Query \times Class \times State \rightarrow Effect \times Errors");
```

SML

```sml
declare_type_abbrev("Ustate", [], Class \times (Effect \times Errors) \times State \rightarrow State \times (Class \times (Data LIST LIST \times Errors)\));
```

We also give an abbreviation definition for the type of state transition functions, \( Stf \).

SML

```sml
declare_type_abbrev("Stf", [], (Query \times Class) \times State \rightarrow State \times (Class \times (Data LIST LIST \times Errors)\));
```

### 3.2 Building a Transition Function

We define a function, \( mkTf \), which builds a transition function from three components: a component of type \( Hide \), a component of type \( Process \) and a component of type \( Ustate \). The resulting transition function updates the original state of the database by using the result of processing a query on the hidden state of the database.

HOL Constant

\[ mkTf \] : \( Hide \rightarrow Process \rightarrow Ustate \rightarrow Stf \)

\[
\forall h : Hide; p : Process; u : Ustate; q : Query; c : Class; s : State
\]

\[
\bullet \quad (mkTf \ h \ p \ u) \ ((q,c),s) = u(c,p(q,c,h(c),s),s)
\]

### 3.3 The SSQL Abstract Machine

An abstract machine is constructed from a pair consisting of a transition function and some initial state, of type \( State : Exp \).

SML

```sml
declare_type_abbrev("Am", [], Stf \times State\);
```

We specify the initial state, \( isstate \) of the SSQL abstract machine.

HOL Constant

\[ isstate \] : State

\[ T \]
An SSQL abstract machine is one which is constructed from the transition function built from the three components \( \text{hide} \), \( \text{processQuery} \) and \( \text{updateState} \), and the initial state \( \text{isstate} \).

\[
\text{HOL Constant}
\]

\[
\text{SSQLam} : Am \\
\text{SSQLam} = (\text{mkTf \ hide \ processQuery \ updateState, isstate})
\]

4 SECURITY CONJECTURE

We formalise the conjecture that the behaviour of the SSQL abstract machine is secure, as defined in [2]. First, we define what we mean by ‘behaviour’.

4.1 Behavioural Abstraction

We define a function \( \text{behaviours} \), generic in QUERY, DATA and STATE, which takes a transition function and a state and yields a behavioural model of type \( \text{BEHAVIOURS} \) (a function from sequences of inputs to sequences of outputs). This behavioural model is independent of the representation of internal states.

We first need to define \( \text{iterate} \) on states in the usual way.

\[
\text{HOL Constant}
\]

\[
\text{iterate} : (((\text{QUERY} \times \text{Class}) \times \text{STATE}) \rightarrow (\text{STATE} \times (\text{Class} \times \text{DATA}))) \\
\rightarrow ((\text{QUERY} \times \text{Class})\text{LIST} \times \text{STATE}) \\
\rightarrow (\text{STATE} \times ((\text{Class} \times \text{DATA})\text{LIST})))
\]

\[
\forall s,\text{STATE}; i,\text{QUERY} \times \text{Class}; si : (\text{QUERY} \times \text{Class})\text{LIST}; \ 
\ 
\text{f} : ((\text{QUERY} \times \text{Class}) \times \text{STATE}) \rightarrow (\text{STATE} \times (\text{Class} \times \text{DATA})) \\
\bullet \ 
(\text{iterate} \ f) ([],s) = (s,[]) \\
\land \\
\text{let} \ 
\text{sso} = (\text{iterate} \ f)(si,s) \ 
\text{in} \\
(\text{iterate} \ f)(si \uparrow [i],s) \\
\ 
= \ 
(\text{Fst}(\text{f}(i,\text{Fst sso})),\text{Snd sso} \uparrow [\text{Snd}(\text{f}(i,\text{Fst sso}))])
\]

We define \( \text{behaviours} \) as an iterated transition function from the initial state:
4.2 SSQL Abstract Machine Correctness Conjecture

Finally, we formally state the correctness conjecture for the SSQL Abstract Machine, i.e. that its behaviour, as defined in section 4.1, is secure, as defined in [2].

\[ \vdash \text{behaviours } \text{SSQLam} \in \text{secure} \]

5 CLOSING DOWN

The following ProofPower instruction restores the previous proof context.

\[ \text{sml} \quad \text{pop}_{-}\text{pc}(); \]
6 THE THEORY fef006

6.1 Parents

fef014

6.2 Children

fef022 fef007

6.3 Constants

\( \text{mkTf} \quad \text{Hide} \rightarrow \text{Process} \rightarrow \text{Ustate} \rightarrow \text{Stf} \)

\( \text{isstate} \quad \text{State} \)

\( \text{SSQLam} \quad \text{Am} \)

\( \text{iterate} \quad ((\text{QUERY} \times \text{Class}) \times \text{'STATE} \rightarrow \text{'STATE} \times \text{Class} \times \text{'DATA}) \rightarrow ((\text{QUERY} \times \text{Class}) \text{LIST} \times \text{'STATE} \rightarrow \text{'STATE} \times (\text{Class} \times \text{'DATA}) \text{LIST} \rightarrow \text{'STATE} \times (\text{Class} \times \text{'DATA}) \text{LIST} \rightarrow ((\text{QUERY} \times \text{Class}) \times \text{'STATE} \rightarrow \text{'STATE} \times \text{Class} \times \text{'DATA}) \times \text{'STATE} \rightarrow ((\text{QUERY}, \text{'DATA}) \text{BEHAVIOIRS}) \)

6.4 Type Abbreviations

\( \text{Hide} \quad \text{Hide} \)

\( \text{Process} \quad \text{Process} \)

\( \text{Ustate} \quad \text{Ustate} \)

\( \text{Stf} \quad \text{Stf} \)

\( \text{Am} \quad \text{Am} \)

6.5 Definitions

\( \text{mkTf} \vdash \forall h \ p \ u \ q \ c \ s \)

\[ \text{mkTf} \ h \ p \ u \ ((q, c), s) = u \ (c, p \ (q, c, h \ (c, s)), s) \]

\( \text{isstate} \vdash \text{true} \)

\( \text{SSQLam} \vdash \text{SSQLam} = (\text{mkTf \ hide \ processQuery \ updateState}, \text{isstate}) \)

\( \text{iterate} \vdash \text{ConstSpec} \)

\[ (\lambda \text{iterate}' \)

\[ \forall s \ i \ si \ f \]

\[ \text{iterate}' \ f \ ([s], s) = (s, []) \]

\[ \wedge (\text{let sso = iterate}' \ f \ (si, s) \)

\[ \text{in iterate}' \ f \ (si \ @ [i], s) \]

\[ = (\text{Fst} \ (f \ i, \text{Fst sso}), \)

\[ \text{Snd sso @ [Snd} \ (f \ i, \text{Fst sso})])]) \)

\[ \text{iterate} \]
behaviours ⊢ ∀ tf istate si
  • behaviours (tf, istate) si
    = Snd (iterate tf (si, istate))
7 INDEX

Am ........................................... 5
behaviours .................................. 7
fef006 .................................... 4
Hide ......................................... 4
isstate ...................................... 5
iterate ....................................... 6
mkTf .......................................... 5
Process ....................................... 5
SSQLam ...................................... 6
Stf ........................................... 5
Ustate ........................................ 5