

Project: DRA FRONT END FILTER PROJECT

Title: Security Conjecture for the SSQL Abstract Machine

Ref: DS/FMU/FEF/006 *Issue: Revision :* 4.3 *Date:* 5 December 2009

Status: Approved *Type:* Specification

Keywords:

Author:

<i>Name</i>	<i>Location</i>	<i>Signature</i>	<i>Date</i>
G. M. Prout	WIN01		

Authorisation for Issue:

<i>Name</i>	<i>Function</i>	<i>Signature</i>	<i>Date</i>
R.B. Jones	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

0 DOCUMENT CONTROL	2
0.1 Contents List	2
0.2 Document Cross References	2
0.3 Changes History	3
0.4 Changes Forecast	3
1 GENERAL	4
1.1 Scope	4
1.2 Introduction	4
2 PRELIMINARIES	4
3 CONSTRUCTION OF AN SSQL ABSTRACT MACHINE	4
3.1 Types of Components	4
3.2 Building a Transition Function	5
3.3 The SSQL Abstract Machine	5
4 SECURITY CONJECTURE	6
4.1 Behavioural Abstraction	6
4.2 SSQL Abstract Machine Correctness Conjecture	7
5 CLOSING DOWN	7
6 THE THEORY fef006	8
6.1 Parents	8
6.2 Children	8
6.3 Constants	8
6.4 Type Abbreviations	9
6.5 Definitions	9
7 INDEX	11

0.2 Document Cross References

- [1] DS/FMU/017. *Secure Database Technical Proposal*. High Assurance Team, ICL Secure Systems, WIN01, 21st January 1992.
- [2] DS/FMU/FEF/003. *Formal Security Policy*. G.M. Prout, ICL Secure Systems, WIN01.
- [3] DS/FMU/FEF/005. *Specifications of hide and updateState*. G.M. Prout, ICL Secure Systems, WIN01.
- [4] DS/FMU/FEF/014. *Specification of SSQL Semantics II*. G.M. Prout, ICL Secure Systems, WIN01.

0.3 Changes History

Issue *Revision : 4.3 (5 December 2009)* Latest approved version.

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, *isstate*. The transition function is built from the three components *hide*, *processQuery* and *updateState*. The function *hide*, defined in [3], operates on the state of the database and returns a ‘sanitised’ version of the state. The function *processQuery*, defined in [4], computes the result of a query on the ‘sanitised’ state of the database. The function *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 *hide* and *updateState*, specified in [3], and *processQuery*, specified in [4]. We first give abbreviation definitions for the types *Hide* of the *hide* component, *Process* of the *processQuery* component and *Ustate* of the *updateState* component.

```
SML
|declare_type_abbrev("Hide",[],[⌈: Class × State → State⌋]);
```

SML

```
| declare_type_abbrev("Process", [], Γ: Query × Class × State → Effect × Errors⊥);
```

SML

```
| declare_type_abbrev("Ustate", [], Γ: Class × (Effect × Errors) × State →  
| State × (Class × (Data LIST LIST × Errors))⊥);
```

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

SML

```
| declare_type_abbrev("Stf", [], Γ: (Query × Class) × State →  
| State × (Class × (Data LIST LIST × 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 → Process → Ustate → Stf
```

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

- $(mkTf\ h\ p\ u)\ ((q,c),s)$
 $=$
 $u(c,p(q,c,h(c,s)),s)$

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

```
| declare_type_abbrev("Am", [], Γ: Stf × 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 *hide*, *processQuery* and *updateState*, and the initial state *isstate*.

HOL Constant

$$\mathbf{SSQLam} : Am$$

$$SSQLam = (mkTf\ hide\ processQuery\ updateState, isstate)$$

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 *behaviours*, generic in QUERY, DATA and STATE, which takes a transition function and a state and yields a behavioural model of type *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 *iterate* on states in the usual way.

HOL Constant

$$\mathbf{iterate} : (((('QUERY \times Class) \times 'STATE) \rightarrow ('STATE \times (Class \times 'DATA))) \rightarrow$$

$$(((('QUERY \times Class)LIST \times 'STATE)$$

$$\rightarrow ('STATE \times ((Class \times 'DATA)LIST)))$$

$$\forall s:'STATE; i:'QUERY \times Class; si:('QUERY \times Class) LIST;$$

$$f: (('QUERY \times Class) \times 'STATE) \rightarrow ('STATE \times (Class \times 'DATA))$$

- $(iterate\ f)\ ([], s) = (s, [])$

$$\wedge$$

$$\text{let } sso = (iterate\ f)(si, s)$$

$$\text{in}$$

$$(iterate\ f)(si \hat{\ } [i], s)$$

$$=$$

$$(Fst(f(i, Fst\ sso)), Snd\ sso \hat{\ } [Snd(f(i, Fst\ sso))])$$

We define *behaviours* as an iterated transition function from the initial state:

HOL Constant

$$\begin{aligned} \mathbf{behaviours} : & (((('QUERY \times Class) \times 'STATE) \\ & \rightarrow ('STATE \times (Class \times 'DATA))) \times 'STATE) \\ & \rightarrow ('QUERY, 'DATA)BEHAVIOURS \end{aligned}$$

$$\forall tf : ('QUERY \times Class) \times 'STATE \rightarrow 'STATE \times (Class \times 'DATA); istate: 'STATE;$$

$$si: ('QUERY \times Class)LIST$$

- $behaviours(tf, istate) \text{ si} = Snd((iterate \text{ tf})(si, istate))$

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].

$$? \vdash \quad behaviours \text{ SSQLam} \in \text{secure}$$

5 CLOSING DOWN

The following ProofPower instruction restores the previous proof context.

SML

$$| \text{pop-pc}();$$

6 THE THEORY fef006

6.1 Parents

fef014

6.2 Children

fef022 fef007

6.3 Constants

mkTf $(Class \times State \rightarrow State)$
 $\rightarrow (Query \times Class \times State$
 $\rightarrow (CHAR\ LIST\ LIST \times (\mathbb{N} \times Data)\ SET\ LIST)$
 $+ (CHAR\ LIST\ LIST \times \mathbb{N}\ SET)$
 $+ (CHAR\ LIST\ LIST$
 $\times (\mathbb{N}$
 $\times (\mathbb{N} \times (ValuedItem + ONE) + Class + Data)$
 $SET) SET)$
 $+ Data\ LIST\ LIST$
 $\times \mathbb{N}\ LIST)$
 $\rightarrow (Class$
 $\times ((CHAR\ LIST\ LIST \times (\mathbb{N} \times Data)\ SET\ LIST)$
 $+ (CHAR\ LIST\ LIST \times \mathbb{N}\ SET)$
 $+ (CHAR\ LIST\ LIST$
 $\times (\mathbb{N}$
 $\times (\mathbb{N}$
 $\times (ValuedItem + ONE)$
 $+ Class$
 $+ Data) SET) SET)$
 $+ Data\ LIST\ LIST$
 $\times \mathbb{N}\ LIST)$
 $\times State$
 $\rightarrow State \times Class \times Data\ LIST\ LIST \times \mathbb{N}\ LIST)$
 $\rightarrow (Query \times Class) \times State$
 $\rightarrow State \times Class \times Data\ LIST\ LIST \times \mathbb{N}\ LIST$

isstate $State$

SSQLam $((Query \times Class) \times State$
 $\rightarrow State \times Class \times Data\ LIST\ LIST \times \mathbb{N}\ LIST)$
 $\times State$

iterate $(('QUERY \times Class) \times 'STATE \rightarrow 'STATE \times Class \times 'DATA)$
 $\rightarrow ('QUERY \times Class)\ LIST \times 'STATE$
 $\rightarrow 'STATE \times (Class \times 'DATA)\ LIST$

behaviours $(('QUERY \times Class) \times 'STATE \rightarrow 'STATE \times Class \times 'DATA)$
 $\times 'STATE$
 $\rightarrow ('QUERY \times Class)\ LIST$
 $\rightarrow (Class \times 'DATA)\ LIST$

6.4 Type Abbreviations

Hide	$Class \times State \rightarrow State$
Process	$Query \times Class \times State$ $\rightarrow (CHAR\ LIST\ LIST \times (\mathbb{N} \times Data)\ SET\ LIST)$ $+ (CHAR\ LIST\ LIST \times \mathbb{N}\ SET)$ $+ (CHAR\ LIST\ LIST$ $\times (\mathbb{N}$ $\times (\mathbb{N} \times (ValuedItem + ONE) + Class + Data)$ $SET) SET)$ $+ Data\ LIST\ LIST$ $\times \mathbb{N}\ LIST$
Ustate	$Class$ $\times ((CHAR\ LIST\ LIST \times (\mathbb{N} \times Data)\ SET\ LIST)$ $+ (CHAR\ LIST\ LIST \times \mathbb{N}\ SET)$ $+ (CHAR\ LIST\ LIST$ $\times (\mathbb{N}$ $\times (\mathbb{N} \times (ValuedItem + ONE) + Class + Data)$ $SET) SET)$ $+ Data\ LIST\ LIST$ $\times \mathbb{N}\ LIST)$ $\times State$ $\rightarrow State \times Class \times Data\ LIST\ LIST \times \mathbb{N}\ LIST$
Stf	$(Query \times Class) \times State$ $\rightarrow State \times Class \times Data\ LIST\ LIST \times \mathbb{N}\ LIST$
Am	$((Query \times Class) \times State$ $\rightarrow State \times Class \times Data\ LIST\ LIST \times \mathbb{N}\ LIST)$ $\times State$

6.5 Definitions

mkTf	$\vdash \forall h\ p\ u\ q\ c\ s$ <ul style="list-style-type: none"> • $mkTf\ h\ p\ u\ ((q, c), s)$ $= u\ (c, p\ (q, c, h\ (c, s)), s)$
isstate	$\vdash true$
SSQLam	$\vdash SSQLam$ $= (mkTf\ hide\ processQuery\ updateState, isstate)$
iterate	$\vdash ConstSpec$ $(\lambda\ iterate'$ <ul style="list-style-type: none"> • $\forall\ s\ i\ si\ f$ • $iterate'\ f\ ([], s) = (s, [])$ $\wedge (let\ sso = iterate'\ f\ (si, s)$ $in\ iterate'\ f\ (si\ @\ [i], s)$ $= (Fst\ (f\ (i, Fst\ sso)),$ $Snd\ sso\ @\ [Snd\ (f\ (i, Fst\ sso))]))))$ $iterate$
behaviours	$\vdash \forall\ tf\ istate\ si$ <ul style="list-style-type: none"> • $behaviours\ (tf, istate)\ si$

$= Snd (iterate\ tf\ (si,\ istate))$

7 INDEX

<i>Am</i>	5
<i>behaviours</i>	7
<i>fef006</i>	4
<i>Hide</i>	4
<i>isstate</i>	5
<i>iterate</i>	6
<i>mkTf</i>	5
<i>Process</i>	5
<i>SSQLam</i>	6
<i>Stf</i>	5
<i>Ustate</i>	5