

Project: DRA FRONT END FILTER PROJECT

Title: Proof Strategy

Ref: DS/FMU/FEF/007

Issue: Revision : 2.3

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: This document contains a proof strategy for proving the SSQL Abstract Machine security conjecture for 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	MAIN SECURITY CONJECTURE	4
4	CRITICAL REQUIREMENTS ON COMPONENTS	4
4.1	Lemma1 : Component Correctness Conjecture	6
5	UNWINDING RESULT	7
5.1	Lemma2	7
5.2	Transition Function Property	7
5.2.1	Lemma3	8
5.3	Iterated Transition Function Property	9
5.3.1	Lemma4	9
5.3.2	Lemma5	10
5.4	Proof of Lemma2, the Unwinding Result	10
6	MAIN PROOF OF SECURITY	11
7	CLOSING DOWN	12
8	THE THEORY fef007	13
8.1	Parents	13
8.2	Children	13
8.3	Constants	13
8.4	Type Abbreviations	13
8.5	Definitions	14
8.6	Theorems	15
9	INDEX	16

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/006. *Security Conjecture for the SSQL Abstract Machine*. G.M. Prout, ICL Secure Systems, WIN01.

0.3 Changes History

Issue Revision : 2.3 (5 December 2009) First approved version.

Issue 2.3 Removed dependency on ICL logo font

0.4 Changes Forecast

None.

1 GENERAL

1.1 Scope

This document provides a proof strategy for proving the security of the SSQL semantics.

1.2 Introduction

In [4], the correctness conjecture for the SSQL Abstract Machine was given. In section 6.2.3 of the Secure Database Technical Proposal, [1], we proposed that a major part of the proof of this conjecture would be an unwinding result independent of the semantics of SSQL. In this document, we provide a proof strategy for the proof of the correctness conjecture for the SSQL Abstract Machine which comprises the proof of an unwinding result together with a proof about the critical components *hide* and *updateState*, defined in [3].

2 PRELIMINARIES

The following ProofPower instructions set up the new theory *fef007*.

SML

```
| open_theory "fef006";
| (force_delete_theory "fef007" handle _ => ());
| new_theory "fef007";
| push_merge_pcs["hol","wrk049","'pair1"]; ;
```

3 MAIN SECURITY CONJECTURE

We reiterate the correctness conjecture for the SSQL Abstract Machine given in [4], i.e. that its behaviour, as defined in [4], is secure, as defined in [2].

```
|      ?⊢      behaviours SSQLam ∈ secure
```

It is the intention to prove this result by proving two main lemmas:

Lemma1 A result about the critical requirements on the components *hide* and *updateState*, formalised in section 4.1.

Lemma2 An unwinding result, formalised in section 5.

4 CRITICAL REQUIREMENTS ON COMPONENTS

We formalise the critical requirements on the critical components *hide* and *updateState* of the SSQL abstract machine. The only requirement exclusively on the *hide* component is that it is monotonic with respect to classification. This is formalised as the property *secureHide*.

HOL Constant

secureHide : *Hide* \mathbb{P}

$$\begin{aligned} &\forall h : \textit{Hide} \bullet \\ &\quad h \in \textit{secureHide} \\ &\quad \Leftrightarrow \\ &\quad \forall c_1 c_2 : \textit{Class}; s_1 s_2 : \textit{State} \bullet \\ &\quad \quad h(c_1, s_1) = h(c_1, s_2) \\ &\quad \quad \wedge \\ &\quad \quad c_1 \textit{ dominates } c_2 \\ &\quad \quad \Rightarrow \\ &\quad \quad h(c_2, s_1) = h(c_2, s_2) \end{aligned}$$

The remaining critical requirements on *hide* and *updateState* are expressed as a critical requirement on the relationship between *hide* and *updateState*. We divide this into four components:

1. If we have two clearances c_1 and c_2 and we update the database at c_1 and a change is visible when hiding w.r.t c_2 , then c_2 must dominate c_1 .
2. If we have two clearances c_1 and c_2 where c_1 dominates c_2 and we update the database at c_2 in states s_1 and s_2 , which are the same when viewed at c_1 , then the resulting states are the same when viewed at c_1 .

These first two requirements concern the security of the state of the database.

3. If we perform an update on two states s_1 and s_2 that are the same when viewed at a clearance c , then the outputs must be the same.
4. Outputs from updates performed at a clearance c must be at clearance c .

These last two requirements concern the security of the outputs from the database to the user.

We define a property *secureUpdate* that captures these requirements

HOL Constant

secureUpdate : $Hide \leftrightarrow Ustate$

$$\begin{aligned} & \forall h : Hide ; u : Ustate \bullet \\ & \quad (h, u) \in secureUpdate \\ & \quad \Leftrightarrow \\ (*1*) & \quad (\forall c_1 c_2 : Class ; s : State ; e : Effect \times Errors \bullet \\ & \quad \quad \text{let } s' = Fst(u(c_1, e, s)) \\ & \quad \quad \text{in} \\ & \quad \quad \quad \neg(h(c_2, s) = h(c_2, s')) \\ & \quad \quad \quad \Rightarrow c_2 \text{ dominates } c_1) \\ & \quad \wedge \\ (*2*) & \quad (\forall c_1 c_2 : Class ; s_1 s_2 : State ; e : Effect \times Errors \bullet \\ & \quad \quad \text{let } s'_1 = Fst(u(c_2, e, s_1)) \\ & \quad \quad \text{and } s'_2 = Fst(u(c_2, e, s_2)) \\ & \quad \quad \text{in} \\ & \quad \quad \quad h(c_1, s_1) = h(c_1, s_2) \\ & \quad \quad \quad \wedge c_1 \text{ dominates } c_2 \\ & \quad \quad \quad \Rightarrow h(c_1, s'_1) = h(c_1, s'_2)) \\ & \quad \wedge \\ (*3*) & \quad (\forall c : Class ; s_1 s_2 : State ; e : Effect \times Errors \bullet \\ & \quad \quad \text{let } o_1 = Snd(u(c, e, s_1)) \\ & \quad \quad \text{and } o_2 = Snd(u(c, e, s_2)) \\ & \quad \quad \text{in} \\ & \quad \quad \quad h(c, s_1) = h(c, s_2) \Rightarrow o_1 = o_2) \\ & \quad \wedge \\ (*4*) & \quad (\forall c : Class ; s : State ; e : Effect \times Errors \bullet \\ & \quad \quad Fst(Snd(u(c, e, s))) = c) \end{aligned}$$

4.1 Lemma1 : Component Correctness Conjecture

We formalise the security conjecture that *hide* and *updateState* are secure.

$$? \vdash \quad hide \in secureHide \wedge (hide, updateState) \in secureUpdate$$

HOL Constant

Lemma1 : *BOOL*

$$\begin{aligned} Lemma1 & = \\ & \quad (hide \in secureHide \\ & \quad \wedge \\ & \quad (hide, updateState) \in secureUpdate) \end{aligned}$$

5 UNWINDING RESULT

5.1 Lemma2

The main unwinding result of the proof of security of the SSQL abstract machine states that if we build an SSQL abstract machine as prescribed in [4] from an *SSQLtf* and an initial state *isstate*, and *hide* and *updateState* satisfy *secureHide* and *secureUpdate*, the security properties on components, then the ‘behaviour’ of the SSQL abstract machine is *secure*, as defined in [2]. This is formalised as *Lemma2*.

$\begin{aligned} & ?\vdash && (hide \in secureHide) \\ & && \wedge \\ & && (hide, updateState) \in secureUpdate) \\ & && \Rightarrow \\ & && behaviours\ SSQLam \in secure \end{aligned}$

HOL Constant

<p>Lemma2 : <i>BOOL</i></p>

<hr style="border: 0.5px solid black;"/> <p><i>Lemma2</i> =</p> $\begin{aligned} & ((hide \in secureHide) \\ & \wedge \\ & (hide, updateState) \in secureUpdate) \\ & \Rightarrow \\ & behaviours\ SSQLam \in secure) \end{aligned}$
--

5.2 Transition Function Property

The critical requirement on *hide* and *updateState* is essentially a requirement on the relationship between *hide* and *updateState*. We require a security property on iterated transition functions in order to prove that the behaviour of the SSQL abstract machine is secure. In turn, we require a security property on transition functions. Hence, in order to prove *Lemma2*, we will prove *Lemma3*, formalised in section 5.2.1, and *Lemma4*, formalised in section 5.3.1, together with *Lemma5*, formalised in section 5.3.2, which is a result relating the iterated SSQL transition function to the behaviour of the SSQL abstract machine. First, a security property on transition functions.

HOL Constant

secureStf : *Stf* \mathbb{P}

$$\begin{aligned}
& \forall stf : Stf \bullet \\
& \quad stf \in secureStf \\
& \quad \Leftrightarrow \\
& \quad (\forall s_1 s_2 : State; i_1 i_2 : Query \times Class; c : Class \bullet \\
& \quad \quad (hide(c, s_1) = hide(c, s_2)) \\
& \quad \quad \wedge \\
& \quad \quad ([i_1], [i_2]) \in same_ins\ c) \\
& \quad \quad \Rightarrow \\
& \quad \quad let \quad (s'_1, o_1) = stf(i_1, s_1) \\
& \quad \quad and \quad (s'_2, o_2) = stf(i_2, s_2) \\
& \quad \quad in \\
& \quad \quad (hide(c, s'_1) = hide(c, s'_2)) \\
& \quad \quad \wedge \\
& \quad \quad ([o_1], [o_2]) \in same_outs\ c)) \\
& \quad \wedge \\
& \quad (\forall s : State; i : Query \times Class; c : Class \bullet \\
& \quad \quad \neg c\ dominates(Snd\ i) \\
& \quad \quad \Rightarrow \\
& \quad \quad ((hide\ (c, s) = hide\ (c, Fst\ (stf\ (i, s)))) \\
& \quad \quad \wedge \\
& \quad \quad \neg c\ dominates(Fst(Snd(stf\ (i, s))))))
\end{aligned}$$

We define a constant *SSQLtf*, the SSQl abstract machine transition function.

HOL Constant

SSQLtf : *Stf*

$$SSQLtf = mkTf\ hide\ processQuery\ updateState$$

5.2.1 Lemma3

We formalise a proposition that states that if *hide* and *updateState* satisfy their critical requirements, then *SSQLtf* satisfies *secureStf*, the security property on transition functions.

$$\begin{aligned}
& ?\vdash \quad (hide \in secureHide) \\
& \quad \wedge \\
& \quad (hide, updateState) \in secureUpdate) \\
& \quad \Rightarrow \\
& \quad SSQLtf \in secureStf
\end{aligned}$$

HOL Constant

Lemma3 : *BOOL*

Lemma3 =
 $((\text{hide} \in \text{secureHide})$
 \wedge
 $(\text{hide}, \text{updateState}) \in \text{secureUpdate})$
 \Rightarrow
 $\text{SSQLtf} \in \text{secureStf}$

5.3 Iterated Transition Function Property

We state a property on iterated transition functions.

First an abbreviation definition for the type of iterated state transition functions, *Itf*.

SML

```
declare_type_abbrev("Itf", [], [·]: (Query × Class)LIST × State
                    → State × (Class × (Data LIST LIST × Errors))LIST∇);
```

HOL Constant

secureItf : *Itf* \mathbb{P}

$\forall \text{itf} : \text{Itf} \bullet$
 $\text{itf} \in \text{secureItf}$
 \Leftrightarrow
 $\forall s_1 s_2 : \text{State}; si_1 si_2 : (\text{Query} \times \text{Class})\text{LIST}; c : \text{Class} \bullet$
 $(\text{hide}(c, s_1) = \text{hide}(c, s_2))$
 \wedge
 $(si_1, si_2) \in \text{same_ins } c$
 \Rightarrow
 $\text{let } (s'_1, so_1) = \text{itf}(si_1, s_1)$
 $\text{and } (s'_2, so_2) = \text{itf}(si_2, s_2)$
 in
 $(\text{hide}(c, s'_1) = \text{hide}(c, s'_2))$
 \wedge
 $(so_1, so_2) \in \text{same_outs } c$

5.3.1 Lemma4

We formalise the proposition that if an *SSQLtf* satisfies *secureStf*, then an iterated *SSQLtf* satisfies *secureItf*, the security property on iterated transition functions.

$$\begin{aligned} ?\vdash \quad & SSQLtf \in secureStf \\ & \Rightarrow \\ & (iterate\ SSQLtf) \in secureItf \end{aligned}$$

HOL Constant

Lemma4 : *BOOL*

$$\begin{aligned} Lemma4 = & \\ & (SSQLtf \in secureStf \\ & \Rightarrow \\ & (iterate\ SSQLtf) \in secureItf) \end{aligned}$$

5.3.2 Lemma5

This proposition states that if the iterated *SSQLtf* satisfies *secureItf*, the security property on iterated transition functions, then the behaviour of the SSQl abstract machine is secure.

$$\begin{aligned} ?\vdash \quad & (iterate\ SSQLtf) \in secureItf \\ & \Rightarrow \\ & behaviours\ SSQLam \in secure \end{aligned}$$

HOL Constant

Lemma5 : *BOOL*

$$\begin{aligned} Lemma5 = & \\ & ((iterate\ SSQLtf) \in secureItf \\ & \Rightarrow \\ & behaviours\ SSQLam \in secure) \end{aligned}$$

5.4 Proof of Lemma2, the Unwinding Result

The proof of the unwinding result, *Lemma2*, follows from the proofs of *Lemma3*, *Lemma4* and *Lemma5*.

We first retrieve the definitions of the lemmas:

SML

```
val Lemma1 = get_specΓ Lemma1∇;
val Lemma2 = get_specΓ Lemma2∇;
val Lemma3 = get_specΓ Lemma3∇;
val Lemma4 = get_specΓ Lemma4∇;
val Lemma5 = get_specΓ Lemma5∇;
```

Then prove *lemma2_thm* which states that if we can prove *Lemma3*, *Lemma4* and *Lemma5*, then we can prove *Lemma2*.

SML

```
| val lemma2_thm = save_thm("lemma2_thm",(prove_rule[Lemma2,Lemma3,Lemma4,Lemma5]
|   "⌈Lemma3 ∧ Lemma4 ∧ Lemma5 ⇒ Lemma2⌋));
```

```
| lemma2_thm = ⊢ Lemma3 ∧ Lemma4 ∧ Lemma5 ⇒ Lemma2
```

6 MAIN PROOF OF SECURITY

The main proof conjecture, given in section 3, that:

```
|   ?⊢   behaviours SSQLam ∈ secure
```

follows from the proofs of *Lemma1*:

```
|   ?⊢   hide ∈ secureHide ∧ (hide,updateState) ∈ secureUpdate
```

and *Lemma2*:

```
|   ?⊢   ((hide ∈ secureHide
|         ∧
|         (hide,updateState) ∈ secureUpdate)
|         ⇒
|         behaviours SSQLam ∈ secure
```

We prove *lemma1_2_thm* which states that if we can prove *Lemma1* and *Lemma2* then we can prove that the behaviour of the SSQL abstract machine is secure.

SML

```
| val lemma1_2_thm = save_thm("lemma1_2_thm",(prove_rule[Lemma1,Lemma2]
|   "⌈Lemma1 ∧ Lemma2 ⇒ behaviours SSQLam ∈ secure⌋));
```

```
| lemma1_2_thm = ⊢ Lemma1 ∧ Lemma2 ⇒ behaviours SSQLam ∈ secure
```

Finally, we prove the *main_thm* that states that if we can prove *Lemma1*, *Lemma3*, *Lemma4* and *Lemma5* then the behaviour of the SSQL abstract machine is secure.

SML

```
| push_goal([],⌈Lemma1 ∧ Lemma3 ∧ Lemma4 ∧ Lemma5 ⇒ behaviours SSQLam ∈ secure⌋);
| a(REPEAT strip_tac THEN MAP_EVERY strip_asm_tac [lemma2_thm,lemma1_2_thm]);
| val main_thm = save_pop_thm"main_thm";
```

```
| main_thm = ⊢ Lemma1 ∧ Lemma3 ∧ Lemma4 ∧ Lemma5 ⇒ behaviours SSQLam ∈ secure
```

7 CLOSING DOWN

The following ProofPower instruction restores the previous proof context.

SML

```
|pop-pc();
```

8 THE THEORY fef007

8.1 Parents

fef006

8.2 Children

fef009

8.3 Constants

secureHide $(Class \times State \rightarrow State) SET$
secureUpdate $((Class \times State \rightarrow State) \times (Class \times ((CHAR LIST LIST \times (\mathbb{N} \times Data) SET LIST) + (CHAR LIST LIST \times \mathbb{N} SET) + (CHAR LIST LIST \times (\mathbb{N} \times (\mathbb{N} \times (ValuedItem + ONE) + Class + Data) SET) SET) + Data LIST LIST \times \mathbb{N} LIST) \times State \rightarrow State \times Class \times Data LIST LIST \times \mathbb{N} LIST)) SET$
Lemma1 $BOOL$
Lemma2 $BOOL$
secureStf $((Query \times Class) \times State \rightarrow State \times Class \times Data LIST LIST \times \mathbb{N} LIST) SET$
SSQLtf $(Query \times Class) \times State \rightarrow State \times Class \times Data LIST LIST \times \mathbb{N} LIST$
Lemma3 $BOOL$
secureItf $((Query \times Class) LIST \times State \rightarrow State \times (Class \times Data LIST LIST \times \mathbb{N} LIST) LIST) SET$
Lemma4 $BOOL$
Lemma5 $BOOL$

8.4 Type Abbreviations

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

8.5 Definitions

secureHide	$\vdash \forall h$ <ul style="list-style-type: none"> • $h \in \text{secureHide}$ $\Leftrightarrow (\forall c_1 c_2 s_1 s_2$ <ul style="list-style-type: none"> • $h(c_1, s_1) = h(c_1, s_2) \wedge c_1 \text{ dominates } c_2$ $\Rightarrow h(c_2, s_1) = h(c_2, s_2))$
secureUpdate	$\vdash \forall h u$ <ul style="list-style-type: none"> • $(h, u) \in \text{secureUpdate}$ $\Leftrightarrow (\forall c_1 c_2 s e$ <ul style="list-style-type: none"> • $(\text{let } s' = \text{Fst}(u(c_1, e, s))$ $\text{in } \neg h(c_2, s) = h(c_2, s')$ $\Rightarrow c_2 \text{ dominates } c_1))$ $\wedge (\forall c_1 c_2 s_1 s_2 e$ <ul style="list-style-type: none"> • $(\text{let } s'_1 = \text{Fst}(u(c_2, e, s_1))$ $\text{and } s'_2 = \text{Fst}(u(c_2, e, s_2))$ $\text{in } h(c_1, s_1) = h(c_1, s_2)$ $\wedge c_1 \text{ dominates } c_2$ $\Rightarrow h(c_1, s'_1) = h(c_1, s'_2))$ $\wedge (\forall c s_1 s_2 e$ <ul style="list-style-type: none"> • $(\text{let } o_1 = \text{Snd}(u(c, e, s_1))$ $\text{and } o_2 = \text{Snd}(u(c, e, s_2))$ $\text{in } h(c, s_1) = h(c, s_2) \Rightarrow o_1 = o_2))$ $\wedge (\forall c s e \bullet \text{Fst}(\text{Snd}(u(c, e, s))) = c)$
Lemma1	$\vdash \text{Lemma1}$ <ul style="list-style-type: none"> $\Leftrightarrow \text{hide} \in \text{secureHide}$ $\wedge (\text{hide}, \text{updateState}) \in \text{secureUpdate}$
Lemma2	$\vdash \text{Lemma2}$ <ul style="list-style-type: none"> $\Leftrightarrow \text{hide} \in \text{secureHide}$ $\wedge (\text{hide}, \text{updateState}) \in \text{secureUpdate}$ $\Rightarrow \text{behaviours SSQLam} \in \text{secure}$
secureStf	$\vdash \forall \text{stf}$ <ul style="list-style-type: none"> • $\text{stf} \in \text{secureStf}$ $\Leftrightarrow (\forall s_1 s_2 i_1 i_2 c$ <ul style="list-style-type: none"> • $\text{hide}(c, s_1) = \text{hide}(c, s_2)$ $\wedge ([i_1], [i_2]) \in \text{same_ins } c$ $\Rightarrow (\text{let } (s'_1, o_1) = \text{stf}(i_1, s_1)$ $\text{and } (s'_2, o_2) = \text{stf}(i_2, s_2)$ $\text{in } \text{hide}(c, s'_1) = \text{hide}(c, s'_2)$ $\wedge ([o_1], [o_2]) \in \text{same_outs } c)$ $\wedge (\forall s i c$ <ul style="list-style-type: none"> • $\neg c \text{ dominates } \text{Snd } i$ $\Rightarrow \text{hide}(c, s) = \text{hide}(c, \text{Fst}(\text{stf}(i, s)))$ $\wedge \neg c \text{ dominates } \text{Fst}(\text{Snd}(\text{stf}(i, s)))$
SSQLtf	$\vdash \text{SSQLtf} = \text{mkTf hide processQuery updateState}$
Lemma3	$\vdash \text{Lemma3}$ <ul style="list-style-type: none"> $\Leftrightarrow \text{hide} \in \text{secureHide}$ $\wedge (\text{hide}, \text{updateState}) \in \text{secureUpdate}$

secureItf $\Rightarrow SSQLtf \in secureStf$
 $\vdash \forall itf$
 • $itf \in secureItf$
 $\Leftrightarrow (\forall s_1 s_2 si_1 si_2 c$
 • $hide(c, s_1) = hide(c, s_2)$
 $\wedge (si_1, si_2) \in same_ins\ c$
 $\Rightarrow (let\ (s'_1, so_1) = itf\ (si_1, s_1)$
 $and\ (s'_2, so_2) = itf\ (si_2, s_2)$
 $in\ hide(c, s'_1) = hide(c, s'_2)$
 $\wedge (so_1, so_2) \in same_outs\ c))$

Lemma4 $\vdash Lemma4$
 $\Leftrightarrow SSQLtf \in secureStf$
 $\Rightarrow iterate\ SSQLtf \in secureItf$

Lemma5 $\vdash Lemma5$
 $\Leftrightarrow iterate\ SSQLtf \in secureItf$
 $\Rightarrow behaviours\ SSQLam \in secure$

8.6 Theorems

lemma2_thm $\vdash Lemma3 \wedge Lemma4 \wedge Lemma5 \Rightarrow Lemma2$

lemma1_2_thm $\vdash Lemma1 \wedge Lemma2 \Rightarrow behaviours\ SSQLam \in secure$

main_thm $\vdash Lemma1 \wedge Lemma3 \wedge Lemma4 \wedge Lemma5$
 $\Rightarrow behaviours\ SSQLam \in secure$

9 INDEX

<i>fef007</i>	4
<i>Itf</i>	9
<i>lemma1_2_thm</i>	11
<i>Lemma1</i>	6
<i>lemma2_thm</i>	11
<i>Lemma2</i>	7
<i>Lemma3</i>	9
<i>Lemma4</i>	10
<i>Lemma5</i>	10
<i>main_thm</i>	11
<i>secureHide</i>	5
<i>secureItf</i>	9
<i>secureStf</i>	8
<i>secureUpdate</i>	6
<i>SSQLtf</i>	8