Project: DRA FRONT END FILTER PROJECT

Title: Execution Model Security Proofs

Ref: DS/FMU/FEF/031 *Issue: Revision* : *1.8* *Date:* 5 June 2016

Status: Draft *Type:* Specification

Keywords:

Author:

Name	*Location*	*Signature*	*Date*
R. D. Arthan	WIN01		

Authorisation for Issue:

Name	*Function*	*Signature*	*Date*
R.B. Jones	HAT Manager		

Abstract: This document contains the formal proofs relating to DS/FMU/FEF/026; it forms part of the Phase II proofs 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 . . . 2
0.4 Changes Forecast . . . 3

1 GENERAL **4**
1.1 Scope . . . 4
1.2 Introduction . . . 4

2 PRELIMINARIES **4**

3 MISCELLANY **4**

4 CONSISTENCY PROOFS **6**

5 EXECUTION MODEL PARTIAL PROOF **7**
5.1 Subsidiary Conjectures and Lemmas . . . 7
5.2 Partial Proof of *EM_SecureE* . . . 10

6 THE THEORY fef031 **11**
6.1 Parents . . . 11
6.2 Children . . . 11
6.3 Constants . . . 11
6.4 Definitions . . . 11
6.5 Theorems . . . 11

7 INDEX **14**

0.2 Document Cross References

[1] DS/FMU/FEF/018. *Proposal for Phase 2.* G.M. Prout, ICL Secure Systems, WIN01.

[2] DS/FMU/FEF/022. *SWORD Front End Architectural Model.* R.D. Arthan, ICL Secure Systems, WIN01.

[3] DS/FMU/FEF/026. *Critical Requirements on the SWORD Query Transformations.* R.D. Arthan, ICL Secure Systems, WIN01.

[4] DS/FMU/FEF/034. *Phase II Proof Strategy.* R.D. Arthan, ICL Secure Systems, WIN01.

0.3 Changes History

Issue *Revision* : *1.8* **(5 June 2016)** *ConditionE_Lemma* now *EM_SecureE*; new boolean constants introduced.

Issue 1.9 Removed dependency on ICL logo font

0.4 Changes Forecast

None.

1 GENERAL

1.1 Scope

This document provides a formal partial proof relating to the specifications in "An Execution Model for SWORD" [3]. It constitutes part of deliverable D13 of work package 3, as given in section 3 of the Proposal for Phase 2, [1].

1.2 Introduction

This document provides a formal partial proof of the conjecture *EM_SecureE* from the proof strategy document, [4], which states that if a compiler and associated database update operation satisfy the *Correct_Compile* correctness criterion of [3], and if the SSQL Query Transformation Processor lies in the set *STP_secure_E* (defined in [3]) determined by the compiler, then the system components satisfy property E of [2] with respect to the representation.

2 PRELIMINARIES

The following ProofPower instructions set up a new theory *fef031* to hold the theorems to be proved and set up a proof context in which to carry out the proofs. The conjecture *EM_SecureE* is defined in theory *fef034* from [4] and so this is also made a parent of the theory *fef031*.

```
SML
open_theory "fef026";
(force_delete_theory "fef031" handle _ => ());
new_theory"fef031";
new_parent"fef034";
set_pc"hol";
```

3 MISCELLANY

```
SML
set_goal([], ⌜∀f g list• Map f (Map g list) = Map (f o g) list⌝);
a(REPEAT strip_tac);
a(list_induction_tac ⌜list⌝ THEN asm_rewrite_tac[get_spec⌜Map⌝]);
val map_o_lemma = save_pop_thm"map_o_lemma";
```

SML

```
set_goal([], ⌜∀c•c lub c = c ∧ lubl[c] = c⌝);
a(strip_tac THEN rewrite_tac(map get_spec[⌜Map⌝, ⌜lubl⌝, ⌜Fold⌝, ⌜Head⌝]));
a(strip_tac);
(* *** Goal "1" *** *)
a(lemma_tac⌜c dominates c ∧ c lub c dominates c⌝ THEN1 prove_tac[get_spec⌜$lub⌝]);
a(all_fc_tac[get_spec⌜$lub⌝]);
a(all_fc_tac[get_spec⌜$lub⌝]);
(* *** Goal "2" *** *)
a(lemma_tac⌜c dominates c lub lattice_bottom ∧ c lub lattice_bottom dominates c⌝
        THEN1 prove_tac[get_spec⌜$lub⌝]);
(* *** Goal "2.1" *** *)
a(lemma_tac⌜c dominates c ∧ c dominates lattice_bottom⌝
        THEN1 prove_tac[get_spec⌜$lub⌝]);
a(all_fc_tac[get_spec⌜$lub⌝]);
(* *** Goal "2.2" *** *)
a(all_fc_tac[get_spec⌜$lub⌝]);
val lub_lemma = save_pop_thm"lub_lemma";
```

SML

```
push_goal([], ⌜∀c1 c2 c3•
        c1 dominates c2 lub c3 ⇔ c1 dominates c2 ∧ c1 dominates c3⌝);
a(REPEAT strip_tac);
(* *** Goal "1" *** *)
a(lemma_tac⌜c2 lub c3 dominates c2⌝ THEN1 rewrite_tac[get_spec⌜$lub⌝]
        THEN all_fc_tac[get_spec⌜$lub⌝]);
(* *** Goal "2" *** *)
a(lemma_tac⌜c2 lub c3 dominates c3⌝ THEN1 rewrite_tac[get_spec⌜$lub⌝]
        THEN all_fc_tac[get_spec⌜$lub⌝]);
(* *** Goal "3" *** *)
a(all_fc_tac[get_spec⌜$lub⌝]);
val dominates_lub_lemma = save_pop_thm"dominates_lub_lemma";
```

SML

```
set_goal([], ⌜∀x1 x2 y1 y2•MkDerTable x1 x2 = MkDerTable y1 y2 ⇔ x1 = y1 ∧ x2 = y2⌝);
a(REPEAT_UNTIL is_⇔ strip_tac THEN ⇔_tac THEN_TRY asm_rewrite_tac[]);
a(LEMMA_T⌜y1 = DT_spec (MkDerTable x1 x2) ∧ y2 = DT_rows (MkDerTable x1 x2)⌝
        rewrite_thm_tac
        THEN1 (asm_rewrite_tac[] THEN rewrite_tac[get_spec⌜MkDerTable⌝]));
a(rewrite_tac[get_spec⌜MkDerTable⌝]);
val MkDerTable_lemma = save_pop_thm"MkDerTable_lemma";
```

SML

```
set_goal([], ⌜∀x1 x2 x3 y1 y2 y3•MkDerTableRow x1 x2 x3 = MkDerTableRow y1 y2 y3
        ⇔ x1 = y1 ∧ x2 = y2 ∧ x3 = y3⌝);
a(REPEAT_UNTIL is_⇔ strip_tac THEN ⇔_tac THEN_TRY asm_rewrite_tac[]);
a(LEMMA_T⌜y1 = DTR_where (MkDerTableRow x1 x2 x3)
        ∧ y2 = DTR_row (MkDerTableRow x1 x2 x3)
         ∧ y3 = DTR_cols (MkDerTableRow x1 x2 x3)⌝
        rewrite_thm_tac
        THEN1 (asm_rewrite_tac[] THEN rewrite_tac[get_spec⌜MkDerTableRow⌝]));
a(rewrite_tac[get_spec⌜MkDerTableRow⌝]);
val MkDerTableRow_lemma = save_pop_thm"MkDerTableRow_lemma";
```

SML

```
push_goal([], ⌜∀upd; qdes•
        Act_t upd qdes
 =      let (query, (dt, errs), st) = qdes
        in
        if      ¬errs = []
        then    (st, ([], errs))
        else if is_select query
        then    (st, (GiveData dt, errs))
        else    (upd (query, (dt, errs), st), ([], []))
⌝);
a(REPEAT strip_tac);
a(lemma_tac⌜∃q d e s•qdes = (q, (d, e), s)⌝
        THEN_LIST[id_tac, asm_rewrite_tac[get_spec⌜Act_t⌝, let_def]]);
a(MAP_EVERY ∃_tac [⌜Fst qdes⌝, ⌜Fst(Fst(Snd qdes))⌝,
        ⌜Snd(Fst(Snd qdes))⌝, ⌜Snd(Snd qdes)⌝]
        THEN rewrite_tac[]);
val Act_t_lemma = save_pop_thm"Act_t_lemma";
```

4 CONSISTENCY PROOFS

SML

```
push_consistency_goal⌜isError⌝;
a(∃_tac⌜(InL, InR, OutL, OutR, IsL, IsR)⌝);
a(rewrite_tac[get_spec⌜IsL⌝]);
a(REPEAT strip_tac THEN_TRY asm_rewrite_tac[]);
(* *** Goal "1" *** *)
a(∃_tac⌜OutL ve⌝ THEN asm_rewrite_tac[]);
(* *** Goal "2" *** *)
a(∃_tac⌜OutR ve⌝ THEN asm_rewrite_tac[]);
val isError_consistency_thm = save_consistency_thm ⌜isError⌝ (pop_thm());
```

5 EXECUTION MODEL PARTIAL PROOF

We introduce two new conjectures that we believe would be expected of the implementation. The first is an information flow property stating that if two *SSQL* states hide the same then their representations as *TSQL* states when viewed as lists of derived tables also hide the same.

HOL Constant

View$_t$_secureE : *BOOL*

$$
\begin{array}{l}
View_t\_secureE \Leftrightarrow \\
\forall c : Class;\ s1\ s2 : State \bullet \\
\quad hide\ (c,\ s1) = hide\ (c,\ s2) \\
\quad \Rightarrow \quad Map\ (HideDerTable\ c)\ (View_t(reprState\ s1)) \\
\quad = \quad Map\ (HideDerTable\ c)\ (View_t(reprState\ s2))
\end{array}
$$

The second is an information flow property on the output filter.

HOL Constant

outputFilter_secureE : *BOOL*

$$
\begin{array}{l}
outputFilter\_secureE \Leftrightarrow \\
\forall q : Query;\ c : Class;\ t1\ t2 : DerTable; \\
\quad dq : Query;\ ocq\text{: } Query + ONE;\ fps : FILTER\_PARS \bullet \\
\quad \neg isError\ (STP\ (q,\ c)) \\
\wedge \quad destVal\ (STP\ (q,\ c)) = (dq,\ ocq,\ fps) \\
\wedge \quad HideDerTable\ c\ t1 = HideDerTable\ c\ t2 \\
\Rightarrow \quad outputFilter(c,\ (GiveData\ t1,\ []),\ fps) \\
\quad = \quad outputFilter(c,\ (GiveData\ t2,\ []),\ fps)
\end{array}
$$

We aim to prove that these two conjectures imply *EM_SecureE*.

5.1 Subsidiary Conjectures and Lemmas

The proof is simplified by the introduction of the following conjectures.

SML

```
val conj100 = ⌜Correct_Compile_STP_secure_E ⇒ Subsys_SecureE⌝;
val conj101 = ⌜∀ compile upd•
      STP ∈ STP_secure_E compile
      ⇒ (EM 1 compile upd, STP, outputFilter) ∈ subsys_secureE reprState⌝;
val conj102 = ⌜∀t : DerTable • GiveData t = [] ⇔ DT_rows t = []⌝;
```

Next, lemmas about these conjectures.

SML

```
set_goal([conj101], conj100);
a(rewrite_tac(map get_spec[⌜Correct_Compile_STP_secure_E⌝,
        ⌜Correct_Compile⌝,⌜Subsys_SecureE⌝]) THEN REPEAT strip_tac);
a(DROP_ASM_T ⌜EM1 compile upd = TSQLtf⌝ (rewrite_thm_tac o eq_sym_rule));
a(asm_fc_tac[] THEN asm_rewrite_tac[]);
val EM_SecureE_Lemma1 = save_pop_thm"EM_SecureE_Lemma1";
```

SML

```
set_goal([⌜View_t_secureE⌝,⌜outputFilter_secureE⌝, conj102], conj101);
a(rewrite_tac(map get_spec[⌜subsys_secureE⌝, ⌜Let⌝,
        ⌜STP_secure_E⌝, ⌜ConditionE⌝]));
a(REPEAT strip_tac);
a(DROP_NTH_ASM_T 5 (strip_asm_tac o
        once_rewrite_rule[prove_rule[]⌜∀p q•p ∨ q ⇔ p ∨ (¬p ∧ q)⌝] o
        list_∀_elim[⌜q⌝, ⌜c⌝]));
(* *** Goal "1" *** *)
a(asm_rewrite_tac(map get_spec[⌜ok_to_proceed⌝]));
(* *** Goal "2" *** *)
a(asm_rewrite_tac(map get_spec[⌜ok_to_proceed⌝, ⌜Let⌝]));
a(all_asm_ante_tac);
a(lemma_tac⌜∃dq ocq fps•destVal (STP (q, c)) = (dq, ocq, fps)⌝
        THEN1
        (MAP_EVERY ∃_tac [⌜Fst (destVal (STP (q, c)))⌝,
                ⌜Fst(Snd (destVal (STP (q, c))))⌝,
                ⌜Snd(Snd (destVal (STP (q, c))))⌝]
        THEN rewrite_tac[]));
a(asm_rewrite_tac[] THEN REPEAT_N 9 strip_tac);
a(lemma_tac⌜View_t (reprState s1) ∈ RiskInputs c (compile dq)⌝);
```

SML

```
(* *** Goal "2.1" *** *)
a(rewrite_tac[get_spec⌜RiskInputs⌝] THEN REPEAT strip_tac);
a(∃_tac⌜View_t (reprState s_2)⌝);
a(DROP_ASM_T ⌜View_t_secureE⌝ (strip_asm_tac o rewrite_rule[get_spec⌜View_t_secureE⌝]));
a(ALL_ASM_FC_T rewrite_tac []);
a(contr_tac);
a(LIST_DROP_NTH_ASM_T [7,8] (MAP_EVERY ante_tac)
        THEN asm_rewrite_tac(Act_t_lemma::map get_spec[⌜EM_1⌝, ⌜EM⌝, ⌜Let⌝]));
a(cases_tac ⌜¬ Snd (compile dq (View_t (reprState s_1))) = []⌝
        THEN cases_tac ⌜is_select dq⌝
        THEN asm_rewrite_tac[]);
(* *** Goal "2.1.1" *** *)
a(contr_tac);
a(DROP_NTH_ASM_T 12 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.1.2" *** *)
a(contr_tac);
a(DROP_NTH_ASM_T 12 ante_tac THEN asm_rewrite_tac[]);
(* *** Goal "2.1.3" *** *)
a(contr_tac);
a(DROP_NTH_ASM_T 12 ante_tac THEN asm_rewrite_tac[]);
a(DROP_ASM_T⌜outputFilter_secureE⌝
        (strip_asm_tac o rewrite_rule[get_spec⌜outputFilter_secureE⌝]));
a(ALL_ASM_FC_T rewrite_tac []);
(* *** Goal "2.1.4" *** *)
a(contr_tac);
a(DROP_NTH_ASM_T 12 ante_tac THEN asm_rewrite_tac[]);
```

SML

```
(* *** Goal "2.2" *** *)
a(LIST_GET_NTH_ASM_T [2] (FC_T (MAP_EVERY ante_tac)));
a(strip_tac THEN asm_rewrite_tac[]);
a(rewrite_tac[taut_rule⌜∀a b c• (a ⇒ ¬b) ⇒ c ⇔ (¬a ⇒ c) ∧ (a ∧ ¬b ⇒ c)⌝]);
a(REPEAT strip_tac);
(* *** Goal "2.2.1" *** *)
a(asm_rewrite_tac(Act_t_lemma::map get_spec[⌜EM_1⌝, ⌜EM⌝, ⌜Let⌝]));
(* *** Goal "2.2.2" *** *)
a(POP_ASM_T ante_tac);
a(asm_rewrite_tac(Act_t_lemma::map get_spec[⌜EM_1⌝, ⌜EM⌝, ⌜Let⌝]));
val EM_SecureE_Lemma2 = save_pop_thm"EM_SecureE_Lemma2";
```

SML

```
set_goal([], conj102);
a(rewrite_tac(map get_spec [⌜GiveData⌝, ⌜Let⌝]));
a(REPEAT strip_tac THEN_TRY asm_rewrite_tac[get_spec⌜Map⌝]);
a(all_asm_ante_tac THEN strip_asm_tac(∀_elim ⌜DT_rows t⌝ list_cases_thm)
        THEN asm_rewrite_tac[get_spec⌜Map⌝]);
val EM_SecureE_Lemma3 = save_pop_thm"EM_SecureE_Lemma3";
```

5.2 Partial Proof of *EM_SecureE*

Finally, we prove that $View_t$_*secureE* and *outputFilter_secureE* imply *EM_SecureE*, the main result about the Execution Model.

SML

```
set_goal([⌜View_t_secureE⌝,⌜outputFilter_secureE⌝],⌜EM_SecureE⌝);
a(rewrite_tac[get_spec⌜EM_SecureE⌝]);
a(REPEAT strip_tac);
a(strip_asm_tac EM_SecureE_Lemma3);
a(strip_asm_tac EM_SecureE_Lemma2);
a(strip_asm_tac EM_SecureE_Lemma1);
val EM_SecureE_thm = save_pop_thm"EM_SecureE_thm";
```

6 THE THEORY fef031

6.1 Parents

fef034 fef026

6.2 Children

fef033

6.3 Constants

View$_t$_secureE
Bool
outputFilter_secureE
Bool

6.4 Definitions

View$_t$_secureE

$$\vdash View_t\_secureE \Leftrightarrow (\forall\ c\ s1\ s2 \bullet hide\ (c, s1) = hide\ (c, s2) \Rightarrow Map\ (HideDerTable\ c)\ (View_t\ (reprState\ s1)) = Map\ (HideDerTable\ c)\ (View_t\ (reprState\ s2)))$$

outputFilter_secureE

$$\vdash outputFilter\_secureE \Leftrightarrow (\forall\ q\ c\ t1\ t2\ dq\ ocq\ fps \bullet \neg\ isError\ (STP\ (q, c)) \wedge destVal\ (STP\ (q, c)) = (dq, ocq, fps) \wedge HideDerTable\ c\ t1 = HideDerTable\ c\ t2 \Rightarrow outputFilter\ (c, (GiveData\ t1, []), fps) = outputFilter\ (c, (GiveData\ t2, []), fps))$$

6.5 Theorems

map_o_lemma $\vdash \forall\ f\ g\ list \bullet Map\ f\ (Map\ g\ list) = Map\ (f\ o\ g)\ list$

lub_lemma $\vdash \forall\ c \bullet c\ lub\ c = c \wedge lubl\ [c] = c$

dominates_lub_lemma

$$\vdash \forall\ c1\ c2\ c3 \bullet c1\ dominates\ c2\ lub\ c3 \Leftrightarrow c1\ dominates\ c2 \wedge c1\ dominates\ c3$$

MkDerTable_lemma

$$\vdash \forall\ x1\ x2\ y1\ y2 \bullet MkDerTable\ x1\ x2 = MkDerTable\ y1\ y2$$

$\Leftrightarrow x1 = y1 \wedge x2 = y2$

MkDerTableRow_lemma

$\vdash \forall\ x1\ x2\ x3\ y1\ y2\ y3$
$\bullet\ MkDerTableRow\ x1\ x2\ x3 = MkDerTableRow\ y1\ y2\ y3$
$\Leftrightarrow x1 = y1 \wedge x2 = y2 \wedge x3 = y3$

Act$_t$_lemma

$\vdash \forall\ upd\ qdes$
$\bullet\ Act_t\ upd\ qdes$
$= (let\ (query, (dt, errs), st) = qdes$
$in\ if\ \neg\ errs = []$
$then\ (st, [], errs)$
$else\ if\ is\_select\ query$
$then\ (st, GiveData\ dt, errs)$
$else\ (upd\ (query, (dt, errs), st), [], []))$

giveVal_consistent
giveError_consistent
destVal_consistent
destError_consistent
isVal_consistent
isError_consistent

$\vdash Consistent$
$(\lambda$
$(giveVal', giveError', destVal', destError',$
$isVal', isError')$
$\bullet \forall\ v\ e\ ve$
$\bullet\ giveVal'\ v = InL\ v$
$\wedge\ giveError'\ e = InR\ e$
$\wedge\ destVal'\ (giveVal'\ v) = v$
$\wedge\ destError'\ (giveError'\ e) = e$
$\wedge\ (isVal'\ ve \Leftrightarrow (\exists\ v_1 \bullet\ ve = giveVal'\ v_1))$
$\wedge\ (isError'\ ve$
$\Leftrightarrow (\exists\ e_1 \bullet\ ve = giveError'\ e_1)))$

EM_SecureE_Lemma1

$\forall\ compile\ upd$
$\bullet\ STP \in STP\_secure\_E\ compile$
$\Rightarrow (EM_1\ compile\ upd, STP, outputFilter)$
$\in subsys\_secureE\ reprState$
$\vdash Correct\_Compile\_STP\_secure\_E \Rightarrow Subsys\_SecureE$

EM_SecureE_Lemma2

$View_t\_secureE,$
$outputFilter\_secureE,$
$\forall\ t \bullet\ GiveData\ t = [] \Leftrightarrow DT\_rows\ t = []$
$\vdash \forall\ compile\ upd$
$\bullet\ STP \in STP\_secure\_E\ compile$
$\Rightarrow (EM_1\ compile\ upd, STP, outputFilter)$
$\in subsys\_secureE\ reprState$

EM_SecureE_Lemma3

$\vdash \forall\ t \bullet\ GiveData\ t = [] \Leftrightarrow DT\_rows\ t = []$

EM_SecureE_thm

$$View_t\_secureE,\ outputFilter\_secureE \vdash EM\_SecureE$$

7 INDEX

Act_t_lemma 6
conj100 7
conj101 7
conj102 7
dominates_lub_lemma 5
EM_SecureE_Lemma1 8
EM_SecureE_Lemma2 9
EM_SecureE_Lemma3 10
EM_SecureE_thm 10
fef031 4
isError_consistency_thm 6
lub_lemma 5
MkDerTableRow_lemma 6
MkDerTable_lemma 5
outputFilter_secureE 7
$View_t$_secureE 7