Project: DRA FRONT END FILTER PROJECT

Title: The Labelling Property for SWORD

Ref: DS/FMU/FEF/043 *Issue: Revision* : *2.1* *Date:* 5 June 2016

Status: Approved *Type:* Specification

Keywords:

Author:

Name	*Location*	*Signature*	*Date*
R. B. Jones	WIN01		

Authorisation for Issue:

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

Abstract: A discussion and formalisation of an abstract 'labelling property' for SWORD. (for DRA Front End Filter project RSRE 1C/6130.)

Distribution: HAT FEF File
Simon Wiseman

0 DOCUMENT CONTROL

0.1 Contents List

0.2 Document Cross References

[1] DS/FMU/FEF/003. *Formal Security Policy.* G.M. Prout, ICL Secure Systems, WIN01.

[2] DS/FMU/FEF/039. *Proposal and Quotation for Phase 3.* R.D. Arthan, ICL Secure Systems, WIN01.

[3] DS/FMU/FEF/040. *Multi-level Formal Security Policy.* R.D. Arthan, ICL Secure Systems, WIN01.

[4] DS/FMU/FEF/042. *Multi-level Architectural Model.* R.D. Arthan, ICL Secure Systems, WIN01.

[5] *Security Properties of the SWORD secure DBMS Design.* Simon Wiseman, DRA.

0.3 Changes History

Issue 1.1 (25 February 1994) First draft.

Issue *Revision* : *2.1* **(5 June 2016)** Final approved version.

Issue 2.2 Removed dependency on ICL logo font

0.4 Changes Forecast

None.

1 GENERAL

1.1 Scope

Phases 1 and 2 of the FEF project dealt exclusively with the 'No Flows Down' security policy as furmulated in [1]. In that formulation, SSQL queries and their results are treated as containing information at a single security classification. The security policy in [1] is extended in [3] and [4] to take account of the multi-level structure of SWORD queries. The result returned to the client from SWORD also consists of a structured multi-level object. However, the way in which the classification fields are to be used in the result from a query, as described in [5, section 2.3], is not consistent with a flow policy which treats these as classification marks in a manner close to their use in the structured multi-level queries. For this reason the multi-level policy, as formulated in [3], treats queries and results differently in expressing the 'No Flows Down' policy. In effect, that policy formulation treats labels other than the outermost label on the result as being insignificant from the point of view of information flow.

In this document we consider extending the policy with a result labelling property which expresses the intended significance of the labels in the result from a SWORD query. The intended significance of these labels in the output is described in [5, section 2.3] and formalisation is attempted in section 4 of the same document. We discuss the problems associated with the formalisation of these additional labelling properties and propose an abstract framework which will permit this kind of property to be expressed.

This document constitutes part of deliverable D17 of work package 7c, as described in section 7 of the Proposal for Phase 3, [2].

1.2 Introduction

The main source of difficulty in arriving at a satisfactory formalisation of the result labelling property is that the description of the required property in [5, section 2.3] explicitly (and we believe essentially) refers to features specific to the query language under consideration. All the formulations of policy matters so far have avoided commitment to the details of the query language, and the complexity of this language would cause a very great impact on the clarity of the policy statement if the policy were to be expressed, as it appears to be informally, in terms of the syntax and semantics of queries.

We therefore propose to define the required property relative to some factorisation of the the semantics of the query language which corresponds to the informal description in an appropriate way, without spelling out the details of this factorisation.

It is also essential that this property be formulated in relation to the pre-filtered outputs of the system, since otherwise the filtering may obscure the information flows and prevent a correct assessment of labels.

1.3 The Specification

An index of the names used in the formal specification may be found in Section 5.

The HOL specification here represents an attempt to capture the key ideas from [5, section 2.3].

1.4 Setting Up

The following ProofPower instructions set up the new theory *fef043* and set the context for the proof tools. The parent theory is the theory *fef042* which contains material about multi-level security policies of the sort we are interested in.

SML

```
open_theory"fef042";
(force_delete_theory "fef043" handle _ => ());
new_theory "fef043";
push_pc "hol";
```

2 OUTPUT FACTORISATION

We now define a 'factorised' output function, which permits the phases in the evaluation of the result of a query to be undertaken separately. This enables an analysis of the flow properties of these separate phases to be used in the formulation of the labelling property.

The stages of a factorisation are intended to correspond to those described in [5, section 2.3]. Conformance of the generalised labelling property given here to the detail of the intended property will depend on the factorisation conforming to the detail of the descriptions in [5, section 2.3].

Each stage of the factorisation computes one level in a four level structured output and provides data for another factor which may be used to compute the next level. After evaluating the first factor we obtain an *Obj* of which the first level is as intended for the final result, and the second level (and below if present) is some representation of the residual query processing to be done by subsequent factors.

After evaluating the second factor on the results of the first factor two layers of data in the resulting *Obj* are as required for the final result. The fourth factor computes values at the fourth level which should contain no further constituent *Obj*s.

In the following type the first *Obj* may be thought of a partially evaluated query and the second a query evaluated one level further.

SML

```
declare_type_abbrev("Factor", ["'State"], ⌜: Class → Obj → 'State → Obj⌝);
```

We now define composition of factors, using which the complete output function can be obtained from a set of factors.

SML

```
declare_infix (250, "⨾f");
```

HOL Constant

$$\$⨾_{\mathbf{f}} : 'State\ Factor \to 'State\ Factor \to 'State\ Factor$$

$$\forall f1\ f2 : 'State\ Factor;\ c : Class;\ s : 'State;\ obj : Obj \bullet$$

$$(f1\ ⨾_f\ f2)\ c\ obj\ s = f2\ c\ (f1\ c\ obj\ s)\ s$$

The informal description in [5, section 2.3] indicates what aspects of the processing of a query are to contribute to the information provided at each level in the structured output. While we do not intend here to fully capture this information, we do express some constraints on which levels of the output of a query are affected by a factor.

An auxiliary used in expressing these contraints is *same_to_level*. This is a relationship between objects indexed by natural numbers, and holds for any index n whenever two objects are identical down (or up!) to level n (down or up from the root, depending on which way you like to visualise a 'tree').

Note that the classification fields are not regarded as classifying themselves and are therefore regarded as being at one level lower than might otherwise have been expected. It is not clear from the sources whether this is intended, but this is consistent with the treatment of multi-level objects in *identicalObj*.

HOL Constant

$$\textbf{same\_to\_level} : \mathbb{N} \rightarrow (Obj \times Obj)\mathbb{P}$$

$$\forall n : \mathbb{N};\ obj1\ obj2 : Obj \bullet$$

$$same\_to\_level\ 0 = \{(obj1, obj2) \mid objectClass\ obj1 = objectClass\ obj2\ \}$$

$$\wedge\ same\_to\_level\ (n+1) = \{ (obj1, obj2) \mid objectContains\ obj1 = objectContains\ obj2 \wedge objectClass\ obj1 = objectClass\ obj2 \wedge \#\ (objectRefers\ obj1) = \#\ (objectRefers\ obj2) \wedge Elems\ (Combine\ (objectRefers\ obj1)\ (objectRefers\ obj2)) \subseteq (same\_to\_level\ n) \}$$

A factor has level n if it changes only information below level n.

HOL Constant

$$\textbf{factor\_level} : \mathbb{N} \rightarrow ('State\ Factor)\mathbb{P}$$

$$\forall n : \mathbb{N} \bullet factor\_level\ n = \{ factor \mid \forall\ c\ state\ obj \bullet (obj, factor\ c\ obj\ state) \in same\_to\_level\ n \}$$

Thus even a level-0 factor may not change the top-level classification of the object.

A factorisation is a collection of four factors:

HOL Labelled Product

Factorisation

$$\textbf{factor0 factor1 factor2 factor3} : 'State\ Factor$$

The factors in a factorisation can be composed as follows:

HOL Constant

$$\textbf{composite} : (' State\ Factorisation) \to ' State\ Factor$$

$$\forall f : ' State\ Factorisation \bullet$$
$$composite\ f = (factor0\ f) \fatsemi_f (factor1\ f) \fatsemi_f (factor2\ f) \fatsemi_f (factor3\ f)$$

A factor can be converted into the output component of a machine as follows:

HOL Constant

$$\textbf{factor\_out} : (' State\ Factorisation) \to ' State \times Req \to Obj$$

$$\forall f : ' State\ Factorisation;\ s : ' State;\ r : Req \bullet$$
$$factor\_out\ f\ (s,r) = (composite\ f)\ (reqClearance\ r)\ (reqSsql\ r)\ s$$

A factorisation is *levelled* if the factors each have appropriate levels:

HOL Constant

$$\textbf{levelled\_factorisation} : (' State\ Factorisation)\mathbb{P}$$

$$\forall facts : ' State\ Factorisation \bullet$$
$$facts \in levelled\_factorisation \Leftrightarrow$$
$$factor0\ facts \in factor\_level\ 0$$
$$\land \quad factor1\ facts \in factor\_level\ 1$$
$$\land \quad factor2\ facts \in factor\_level\ 2$$
$$\land \quad factor3\ facts \in factor\_level\ 3$$

For a factorisation to be correct it must also fail to pre-evaluate levels in the query (without this constraint the first factor could fully evaluate the query and subsequent factors could be identity functions). However, this detail can only be specified by reference to the form of a query in greater detail than is considered appropriate here.

A factorisation factors a particular machine if, when the factors are put together, the resulting output corresponds to that of the machine:

A *FactoredMachine* is a *Machine* together with a factorisation of that machine.

HOL Labelled Product

FactoredMachine

machine	: $' State\ Machine$;
factors	: $' State\ Factorisation$

A factored machine is *well_factored* if the factors form a *levelled_factorisation* and the output function determined by the factors is the same as that in the machine.

HOL Constant

well_factored : $('State\ FactoredMachine)\mathbb{P}$

$\forall m :' State\ FactoredMachine \bullet$
$m \in well\_factored \Leftrightarrow$
$\quad (factors\ m) \in levelled\_factorisation$
$\wedge \quad Output\ (machine\ m) = factor\_out\ (factors\ m)$

3 THE LABELLING PROPERTY

The labelling property is formalised as follows. For any input history and single request we consider the behaviour of the machine. A separate property is required for label correctness at each level in the output. The property for level n is obtained by constructing a machine which maps input histories to outputs and then asserting the information flow security of that machine. These machines will be constructed so that they first operate in the normal way on the input history to determine the state in which the query under consideration will be evaluated. The *partially* evaluated query will then be evaluated to completion in this state. The effect of this construction is to ignore the information flows which take place in the initial stage of the query and confine the security constraint to the information flows occuring in the subsequent evaluation of the query.

For each combination of:

- a Machine (m)
- a binary factorisation of the machine output function (*f1, f2*)
- a request list (which determines a state) (*rl*)
- a request (submitted in that state) (r)

we construct a *special_machine*.

The special machine will have pre-compiled into it the result of evaluating the first factor (*f1*) on the given request (r) in the state determined by the request list (*rl1*). It will complete the evaluation (using *f2*) in a possibly different state, as determined by some other request list (*rl2*), and return the resulting output as an *Obj*.

HOL Constant

special_machine : $('State\ Machine \times 'State\ Factor \times 'State\ Factor)$
$\quad \rightarrow Req\ LIST \rightarrow Req \rightarrow (Req\ LIST \rightarrow Obj)$

$\forall (m, f1, f2);\ rl1\ rl2 : Req\ LIST;\ r : Req \bullet$
$special\_machine\ (m, f1, f2)\ rl1\ r\ rl2 =$
$\quad let\ s1 = Snd\ (lift\_machine\ m\ (Init\ m, rl1))$
$\quad and\ s2 = Snd\ (lift\_machine\ m\ (Init\ m, rl2))$
$\quad in \quad let\ pe\_req = f1\ (reqClearance\ r)\ (reqSsql\ r)\ s1$
$\quad\quad in\ f2\ (reqClearance\ r)\ pe\_req\ s2$

To express the flow contraints required on these special machines we need a sameness relation which concerns itself only with information below a certain level in the output object.

HOL Constant

$$\textbf{same\_at\_c\_below\_level} : \mathbb{N} \to \mathit{Class} \to (\mathit{Obj} \leftrightarrow \mathit{Obj})$$

$$\begin{array}{ll} & \forall n : \mathbb{N};\ c : \mathit{Class};\ \mathit{obj1}\ \ \mathit{obj2} : \mathit{Obj} \bullet \\ & \\ & \mathit{same\_at\_c\_below\_level}\ 0\ c = \mathit{identicalObj}\ c \\ \wedge & \mathit{same\_at\_c\_below\_level}\ (n{+}1)\ c = \\ & \quad \{ \quad (\mathit{obj1},\ \mathit{obj2}) \\ & \quad \mid \quad \#\ (\mathit{objectRefers}\ \mathit{obj1}) = \#\ (\mathit{objectRefers}\ \mathit{obj2}) \\ & \quad \wedge \quad \mathit{Elems}\ (\mathit{Combine}\ (\mathit{objectRefers}\ \mathit{obj1})\ (\mathit{objectRefers}\ \mathit{obj2})) \\ & \qquad \subseteq (\mathit{same\_at\_c\_below\_level}\ n\ c) \\ & \quad \} \end{array}$$

A binary factorisation of a machine is then *label_secure_to* some particular level if the special machine constructed using that binary factorisation is always *ml_secure* relative to sameness below level n.

HOL Constant

$$\textbf{label\_secure\_to} : \mathbb{N} \to (\mathit{'State\ Machine} \times \mathit{'State\ Factor} \times \mathit{'State\ Factor})\ \mathbb{P}$$

$$\begin{array}{ll} \forall\ n : \mathbb{N} \bullet & \mathit{label\_secure\_to}\ n = \\ & \{ \quad \mathit{mff} \\ & \mid \quad \forall\ \mathit{rl} : \mathit{Req\ LIST};\ r : \mathit{Req} \bullet \\ & \quad \mathit{let}\ \mathit{sm} = \mathit{special\_machine}\ \mathit{mff}\ \mathit{rl}\ r \\ & \quad \mathit{in}\ \mathit{sm} \in \mathit{ml\_secure}\ \mathit{sameRequests}\ (\mathit{same\_at\_c\_below\_level}\ n) \\ & \} \end{array}$$

A fully factorised machine is *label_secure* if the three binary factorisations which correspond to partial evaluation to levels 1, 2 and 3 are all *label_secure_n* at the appropriate levels.

HOL Constant

$$\textbf{label\_secure} : (\mathit{'State\ FactoredMachine})\ \mathbb{P}$$

$$
\begin{array}{l}
\mathit{label\_secure} = \\
\quad \{ \quad \mathit{fm} : \mathit{'State\ FactoredMachine} \\
\quad | \quad \mathit{let}\ m = \mathit{machine}\ \mathit{fm}\ \mathit{and}\ \mathit{fs} = \mathit{factors}\ \mathit{fm} \\
\qquad \mathit{in} \\
\qquad \mathit{let}\ \mathit{lf1} = (\mathit{factor0}\ \mathit{fs}) \\
\qquad \mathit{and}\ \mathit{lf2} = (\mathit{factor0}\ \mathit{fs}) \fatsemi_f (\mathit{factor1}\ \mathit{fs}) \\
\qquad \mathit{and}\ \mathit{lf3} = (\mathit{factor0}\ \mathit{fs}) \fatsemi_f (\mathit{factor1}\ \mathit{fs}) \fatsemi_f (\mathit{factor2}\ \mathit{fs}) \\
\qquad \mathit{and}\ \mathit{rf1} = (\mathit{factor1}\ \mathit{fs}) \fatsemi_f (\mathit{factor2}\ \mathit{fs}) \fatsemi_f (\mathit{factor3}\ \mathit{fs}) \\
\qquad \mathit{and}\ \mathit{rf2} = (\mathit{factor2}\ \mathit{fs}) \fatsemi_f (\mathit{factor3}\ \mathit{fs}) \\
\qquad \mathit{and}\ \mathit{rf3} = (\mathit{factor3}\ \mathit{fs}) \\
\qquad \mathit{in} \quad (m, \mathit{lf1}, \mathit{rf1}) \in \mathit{label\_secure\_to}\ 1 \\
\qquad\quad \wedge \quad (m, \mathit{lf2}, \mathit{rf2}) \in \mathit{label\_secure\_to}\ 2 \\
\qquad\quad \wedge \quad (m, \mathit{lf3}, \mathit{rf3}) \in \mathit{label\_secure\_to}\ 3 \\
\quad \}
\end{array}
$$

4 CLOSING DOWN

The following ProofPower instruction restores the previous proof context.

SML

```
pop_pc();
```

5 INDEX