Project:	DRA FRONT END FILTER PROJECT		
Title:	Formal Security Policy		
Ref:	DS/FMU/FEF/003	*Issue: Revision : 5.5*	*Date:* 5 June 2016
Status:	Draft		*Type:* Specification

Keywords:

Author:

Name	*Location*	*Signature*	*Date*
G. M. Prout	WIN01		

Authorisation for Issue:

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

Abstract: The formalisation of the 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

0.2 Document Cross References

[1] DS/FMU/017. *Secure Database Technical Proposal.* High Assurance Team, ICL Secure Systems, WIN01, 21st January 1992.

[2] *Invitation to Tender RSRE 1c/6130.* DRA, December 1991.

0.3 Changes History

Issue *Revision : 5.5* **(5 June 2016)** Added *glb*.

Issue 5.6 Removed dependency on ICL logo font

0.4 Changes Forecast

None.

1 GENERAL

1.1 Scope

This document gives a formal specification of a behavioural model of systems and a formal specification of a security policy which is a property on such systems. It constitutes deliverable D1 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 use a formalisation of a policy which is essentially a *non-interference* formulation of the *no flows down* requirement expressed in Annex 1, *The SWORD Secure DBMS*, to the ITT [2]. The formal material from Section 5 of [1] has been transferred to this document. This comprises a formal specification of a behavioural model of systems and a formal specification of a security policy which is a property on such systems. The SSQL specifications will be formalised as a particular behavioural model so that it can be proven that SSQL provides information flow security.

2 FORMAL SECURITY POLICY

2.1 Setting Up

The following ProofPower instructions set up the new theory *fef003* and set the context for the proof tools.

SML

```
open_theory"wrk049";
(force_delete_theory "fef003" handle _ => ());
new_theory "fef003";
push_pc "hol";
```

An index of the names used in the formal specification may be found in Section 6. A listing of the theory *fef003* created by processing this document using ProofPower may be found at the end of this document.

2.2 Classification

We begin formalising the security model by defining a relation *dominates* on classifications. We introduce a new type to represent the classifications.

SML

```
new_type("Class",0);
```

Classifications are partially ordered by the relation *dominates*. We define *lattice_bottom*, *lattice_top*, the a least upper bound on classes, *lub* and the greatest lower bound on classes, *glb*.

SML

```
declare_infix (150,"dominates");
declare_infix (250,"lub");
declare_infix (250,"glb");
```

HOL Constant

$$
\begin{array}{l}
\$\mathbf{dominates} : Class \to Class \to BOOL;\\
\$\mathbf{lattice\_bottom} : Class;\\
\$\mathbf{lattice\_top} : Class;\\
\$\mathbf{lub} : Class \to Class \to Class;\\
\$\mathbf{glb} : Class \to Class \to Class
\end{array}
$$

$$
\begin{array}{lll}
& \forall\ x\ y\ z{:}\ Class \bullet & \\
& & x\ dominates\ x\\
& \wedge & (x\ dominates\ y \wedge y\ dominates\ x \Rightarrow x = y)\\
& \wedge & (x\ dominates\ y \wedge y\ dominates\ z \Rightarrow x\ dominates\ z)\\
\wedge & (\forall\ x \bullet x\ dominates\ lattice\_bottom) & \\
\wedge & (\forall\ x \bullet lattice\_top\ dominates\ x) & \\
\wedge & (\forall\ x\ y\ z \bullet (x\ lub\ y\ dominates\ x) & \\
& & \wedge\ (x\ lub\ y\ dominates\ y)\\
& & \wedge\ (z\ dominates\ x \wedge z\ dominates\ y \Rightarrow z\ dominates\ x\ lub\ y))\\
\wedge & (\forall\ x\ y\ z \bullet (x\ dominates\ x\ glb\ y) & \\
& & \wedge\ (y\ dominates\ x\ glb\ y)\\
& & \wedge\ (x\ dominates\ z \wedge y\ dominates\ z \Rightarrow x\ glb\ y\ dominates\ z))
\end{array}
$$

2.3 The Type of Behavioural Models of Systems

The definition of information flow security presented here is a property on behavioural models of systems. An input to a system under consideration is a single query and a classification. Output from the system generated by a single query is a classification and some data.

A 'behaviour' of a system is a single sequence of inputs and the resulting sequence of outputs. A behavioural model is the set of all possible behaviours. This is formalised as a function from the sequence of permissible inputs to the system, yielding the corresponding outputs. The behavioural model is independent of the state of the system and generic in QUERY and DATA.

SML

```
declare_type_abbrev("BEHAVIOURS",["'QUERY","'DATA"],
        ⌜: ('QUERY × Class)LIST → (Class × 'DATA)LIST⌝);
```

2.4 Critical Requirements

We now attempt to capture (in fact *define*) what it means to say that such a behavioural model is *secure*. The critical requirement is then that the behavioural model be secure.

The intended meaning of *secure* here concerns the nature of the information flows permitted by the behavioural model.

The formulation below is an "interference style" formulation. To express the flow constraint in this way, it is first necessary to define filtering operations on the inputs and outputs of the system.

Two sequences of inputs *si1* and *si2* are the same when viewed from a classification *clear* if, after purging from both any input whose classification is not dominated by *clear*, the two resulting purged sequences are identical.

HOL Constant

$$\textbf{same\_ins} : Class \to (('QUERY \times Class)LIST \leftrightarrow ('QUERY \times Class)LIST)$$

$$\begin{array}{ll} \forall clear: Class;\ si1\ si2: ('QUERY \times Class)LIST & \\ \bullet & (si1,si2) \in same\_ins\ clear \\ & \Leftrightarrow \\ & let\ v = \{(q,c) | (clear\ dominates\ c)\} \\ & in \\ & si1 \upharpoonright v = si2 \upharpoonright v \end{array}$$

Two sequences of outputs *so1* and *so2* are the same when viewed from a classification *clear* if, after purging from both any output whose classification is not dominated by *clear*, the two resulting purged sequences are identical.

HOL Constant

$$\textbf{same\_outs} : Class \to ((Class \times 'DATA)LIST \leftrightarrow (Class \times 'DATA)LIST)$$

$$\begin{array}{ll} \forall clear: Class;\ so1\ so2: (Class \times 'DATA)LIST & \\ \bullet & (so1,so2) \in same\_outs\ clear \\ & \Leftrightarrow \\ & let\ v = \{(c,d) | (clear\ dominates\ c)\} \\ & in \\ & so1 \upharpoonright v = so2 \upharpoonright v \end{array}$$

The constraint on information flows proposed for verification is that outputs classified at classes dominated by any class *clear* are independent of inputs at classifications which are not dominated by *clear*. We assume that mechanisms outside the scope of this model ensure that users see only those outputs which they are cleared to see, and that inputs are correctly classified.

This is expressed by saying that if two sequences of inputs are the same when viewed at a certain classification then the outputs will be the same when viewed at that classification.

HOL Constant

$$\textbf{secure}: ('QUERY,'DATA)BEHAVIOURS\ \mathbb{P}$$

$$\forall bm{:}('QUERY,'DATA)BEHAVIOURS \bullet\ bm \in secure \Leftrightarrow \forall\ clear : Class;\ si1\ si2 :('QUERY \times Class)LIST \bullet\ (si1,si2) \in same\_ins\ clear \Rightarrow (bm\ si1,bm\ si2) \in same\_outs\ clear$$

The enunciation of the security requirement as a *property* of behavioural models of systems enables us to express directly the claim that a behavioural model of an implementation is secure. The formal specifications of the design and implementation of the SSQL system will then be expressed as entities, the behavioural model of which has type (some instance of) *BEHAVIOURS*. The proposition that they exhibit secure behaviour will then be expressible, and provable.

3 CLOSING DOWN

The following ProofPower instruction restores the previous proof context.

SML

```
pop_pc();
```

4 THE SECURITY MODEL JUSTIFICATION

The formal security policy given here is exactly that of the Secure Database Technical Proposal, [1], which is the basis of the current contract. After reconsideration, it was decided that the constraints on the function *BEHAVIOURS* should be removed since they were not part of the security policy and hence were irrelevant.

The justification for using *non-interference* as a security model for the SSQL front end filter may be found in the text introducing and defining *same_ins*, *same_outs* and *secure* in Section 2.4. This is a straightforward formalisation of the *no flows down* policy cited in Annex 1 of the ITT [2] that results returned to clients are affected only by inputs of lower or equal classification.

We have defined information flow security as a property on behavioural models of systems. This means that security is dependent only on inputs to and outputs from the system, and is independent of the state of the system. A secure refinement from the SSQL abstract machine to an SSQL implementation is one which preserves behaviour, hence we may refine the state of the system to an implementation provided we maintain its behaviour.

5 THE THEORY fef003

5.1 Parents

wrk049

5.2 Children

fef040 fef004

5.3 Constants

$glb	$Class \to Class \to Class$
$lub	$Class \to Class \to Class$
lattice_top	$Class$
lattice_bottom	$Class$
$dominates	$Class \to Class \to BOOL$
same_ins	$Class \to ('QUERY \times Class)\ LIST \leftrightarrow ('QUERY \times Class)\ LIST$
same_outs	$Class \to (Class \times 'DATA)\ LIST \leftrightarrow (Class \times 'DATA)\ LIST$
secure	$('QUERY, 'DATA)\ BEHAVIOURS\ \mathbb{P}$

5.4 Types

Class

5.5 Type Abbreviations

$('QUERY, 'DATA)$ **BEHAVIOURS**
$('QUERY, 'DATA)\ BEHAVIOURS$

5.6 Fixity

Right Infix 150:
dominates

Right Infix 250:
glb **lub**

5.7 Definitions

dominates
lattice_bottom
lattice_top
lub
glb $\vdash$ *ConstSpec*

$$
\begin{array}{l}
(\lambda \\
\quad (dominates', lattice\_bottom', lattice\_top', \\
\quad\quad lub', glb') \\
\quad \bullet \forall\ x\ y\ z \\
\quad\quad \bullet\ dominates'\ x\ x \\
\quad\quad\quad \wedge (dominates'\ x\ y \wedge dominates'\ y\ x \Rightarrow x = y) \\
\quad\quad\quad \wedge (dominates'\ x\ y \wedge dominates'\ y\ z \\
\quad\quad\quad\quad \Rightarrow dominates'\ x\ z) \\
\quad\quad\quad \wedge (\forall\ x \bullet\ dominates'\ x\ lattice\_bottom') \\
\quad\quad\quad \wedge (\forall\ x \bullet\ dominates'\ lattice\_top'\ x) \\
\quad\quad\quad \wedge (\forall\ x\ y\ z \\
\quad\quad\quad \bullet\ dominates'\ (lub'\ x\ y)\ x \\
\quad\quad\quad\quad \wedge\ dominates'\ (lub'\ x\ y)\ y \\
\quad\quad\quad\quad \wedge (dominates'\ z\ x \wedge dominates'\ z\ y \\
\quad\quad\quad\quad\quad \Rightarrow dominates'\ z\ (lub'\ x\ y))) \\
\quad\quad\quad \wedge (\forall\ x\ y\ z \\
\quad\quad\quad \bullet\ dominates'\ x\ (glb'\ x\ y) \\
\quad\quad\quad\quad \wedge\ dominates'\ y\ (glb'\ x\ y) \\
\quad\quad\quad\quad \wedge (dominates'\ x\ z \wedge dominates'\ y\ z \\
\quad\quad\quad\quad\quad \Rightarrow dominates'\ (glb'\ x\ y)\ z))) \\
(\$dominates, lattice\_bottom, lattice\_top, \$lub, \\
\quad \$glb)
\end{array}
$$

same_ins $\vdash \forall\ clear\ si1\ si2$

$$
\begin{array}{l}
\bullet\ (si1, si2) \in same\_ins\ clear \\
\quad \Leftrightarrow (let\ v = \{(q, c) | clear\ dominates\ c\} \\
\quad\quad in\ si1 \upharpoonright v = si2 \upharpoonright v)
\end{array}
$$

same_outs $\vdash \forall\ clear\ so1\ so2$

$$
\begin{array}{l}
\bullet\ (so1, so2) \in same\_outs\ clear \\
\quad \Leftrightarrow (let\ v = \{(c, d) | clear\ dominates\ c\} \\
\quad\quad in\ so1 \upharpoonright v = so2 \upharpoonright v)
\end{array}
$$

secure $\vdash \forall\ bm$

$$
\begin{array}{l}
\bullet\ bm \in secure \\
\quad \Leftrightarrow (\forall\ clear\ si1\ si2 \\
\quad \bullet\ (si1, si2) \in same\_ins\ clear \\
\quad\quad \Rightarrow (bm\ si1, bm\ si2) \in same\_outs\ clear)
\end{array}
$$

6 INDEX