Abstract: The formalisation of the security conjecture for the DRA front end filter project RSRE 1C/6130.
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 FORMAL SECURITY POLICY ........................................... 4
  2.1 Setting Up ............................................................. 4
  2.2 Classification ......................................................... 4
  2.3 The Type of Behavioural Models of Systems .................. 5
  2.4 Critical Requirements ............................................. 5

3 CLOSING DOWN ............................................................ 7

4 THE SECURITY MODEL JUSTIFICATION ........................... 7

5 THE THEORY fef003 ...................................................... 8
  5.1 Parents .................................................................. 8
  5.2 Children ............................................................... 8
  5.3 Constants ............................................................. 8
  5.4 Types .................................................................... 8
  5.5 Type Abbreviations ................................................ 8
  5.6 Fixity .................................................................... 8
  5.7 Definitions ............................................................ 9

6 INDEX ............................................................................. 10

0.2 Document Cross References


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.
Lemma 1

Formal Security Policy

Ref: DS/FMU/FEF/003

Issue: Revision: 5.5

Date: 5 June 2016

SML

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

HOL Constant

|$\text{dominates} : \text{Class} \rightarrow \text{Class} \rightarrow \text{BOOL};$
|$\text{lattice\_bottom} : \text{Class};$
|$\text{lattice\_top} : \text{Class};$
|$\text{lub} : \text{Class} \rightarrow \text{Class} \rightarrow \text{Class};$
|$\text{glb} : \text{Class} \rightarrow \text{Class} \rightarrow \text{Class} |

\[
\forall x y z : \text{Class} \bullet \\
\quad x \text{ dominates } x \\
\quad \land (x \text{ dominates } y \land y \text{ dominates } x \Rightarrow x = y) \\
\quad \land (x \text{ dominates } y \land y \text{ dominates } z \Rightarrow x \text{ dominates } z) \\
\quad \land (\forall x \bullet x \text{ dominates lattice\_bottom}) \\
\quad \land (\forall x \bullet \text{lattice\_top dominates } x) \\
\quad \land (\forall x y z \bullet (x \text{ lub } y \text{ dominates } x) \\
\quad \land (x \text{ lub } y \text{ dominates } y) \\
\quad \land (z \text{ dominates } x \land z \text{ dominates } y \Rightarrow z \text{ dominates } x \text{ lub } y)) \\
\quad \land (\forall x y z \bullet (x \text{ dominates } x \text{ glb } y) \\
\quad \land (y \text{ dominates } x \text{ glb } y) \\
\quad \land (x \text{ dominates } z \land y \text{ dominates } z \Rightarrow x \text{ glb } y \text{ dominates } z))
\]

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"]). |
| $\Gamma : (\langle QUERY \times Class\rangle \text{LIST} \rightarrow (\text{Class} \times \langle DATA\rangle \text{LIST}));$ |

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 $s_{i1}$ and $s_{i2}$ 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**

```
**same_ins** : Class $\rightarrow$ (("QUERY $\times$ Class")$LIST \leftrightarrow$ (("QUERY $\times$ Class")$LIST
```

\[
\forall clear \, : \, Class; \, si1, si2 \, : \, ("QUERY $\times$ Class")$LIST
\]

\[
\Rightarrow
\]

\[
let \, v = \{(q,c)|(clear \, dominates \, c)\}
\]

\[
in
\]

\[
si1 \downarrow v = si2 \downarrow v
\]

Two sequences of outputs $s_{o1}$ and $s_{o2}$ 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**

```
**same_outs** : Class $\rightarrow$ ((Class $\times$ 'DATA')$LIST \leftrightarrow$ (Class $\times$ 'DATA')$LIST
```

\[
\forall clear \, : \, Class; \, so1, so2 \, : \, (Class $\times$ 'DATA')$LIST
\]

\[
\Rightarrow
\]

\[
let \, v = \{(c,d)|(clear \, dominates \, c)\}
\]

\[
in
\]

\[
so1 \downarrow v = so2 \downarrow v
\]

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.
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

\texttt{wrk049}

5.2 Children

\texttt{fef040 fef004}

5.3 Constants

\begin{itemize}
  \item \texttt{\$\textit{glb}} \quad \text{\texttt{Class \rightarrow Class \rightarrow Class}}
  \item \texttt{\$\textit{lub}} \quad \text{\texttt{Class \rightarrow Class \rightarrow Class}}
  \item \texttt{\textit{lattice\_top}} \quad \text{\texttt{Class}}
  \item \texttt{\textit{lattice\_bottom}} \quad \text{\texttt{Class}}
  \item \texttt{\$\textit{dominates}} \quad \text{\texttt{Class \rightarrow Class \rightarrow BOOL}}
  \item \texttt{\textit{same\_ins}} \quad \text{\texttt{Class}}
    \rightarrow (\texttt{\textquotesingle QUERY} \times \text{\texttt{Class}}) \text{\texttt{LIST}} \leftrightarrow (\texttt{\textquotesingle QUERY} \times \text{\texttt{Class}}) \text{\texttt{LIST}}
  \item \texttt{\textit{same\_outs}} \quad \text{\texttt{Class \rightarrow (Class \times \texttt{\textquotesingle DATA}) \texttt{LIST}}} \leftrightarrow (\texttt{\texttt{\textquotesingle DATA}} \times \text{\texttt{\textquotesingle DATA}) \text{\texttt{LIST}}}
  \item \texttt{\textit{secure}} \quad \text{\texttt{(\textquotesingle QUERY, \texttt{\textquotesingle DATA) BEHAVIOURS}} \texttt{F}}
\end{itemize}

5.4 Types

\texttt{Class}

5.5 Type Abbreviations

\begin{itemize}
  \item \texttt{(\textquotesingle QUERY, \texttt{\textquotesingle DATA) BEHAVIOURS}}
  \item \texttt{(\textquotesingle QUERY, \texttt{\textquotesingle DATA) BEHAVIOURS}}
\end{itemize}

5.6 Fixity

Right Infix 150:

\begin{itemize}
  \item \texttt{\textit{dominates}}
\end{itemize}

Right Infix 250:

\begin{itemize}
  \item \texttt{\textit{glb} lub}
\end{itemize}
5.7 Definitions

domines
lattice_bottom
lattice_top
lub
glb

⊢ ConstSpec
(\lambda
  (dominates', lattice_bottom', lattice_top',
  lub', glb')
  \forall x y z
  • dominates' x x
  \land (dominates' x y \land dominates' y x \implies x = y)
  \land (dominates' x y \land dominates' y z
  \implies dominates' x z)
  \land (\forall z \bullet dominates' x lattice_bottom')
  \land (\forall x \bullet dominates' lattice_top' x)
  \land (\forall x y z
  • dominates' (lub' x y) x
  \land dominates' (lub' x y) y
  \land (dominates' z x \land dominates' z y
  \implies dominates' z (lub' x y)))
  \land (\forall x y z
  • dominates' x (glb' x y)
  \land dominates' y (glb' x y)
  \land (dominates' x z \land dominates' y z
  \implies dominates' (glb' x y) z)))

($dominates, lattice_bottom, lattice_top, $lub, $glb$

same_ins ⊢ ∀ clear si1 si2
  • (si1, si2) ∈ same_ins clear
    ⇔ (let v = {(q, c)|clear dominates c}
    in si1 | v = si2 | v)

same_outs ⊢ ∀ clear so1 so2
  • (so1, so2) ∈ same_outs clear
    ⇔ (let v = {(c, d)|clear dominates c}
    in so1 | v = so2 | v)

secure ⊢ ∀ bm
  • bm ∈ secure
    ⇔ (\forall clear si1 si2
    • (si1, si2) ∈ same_ins clear
      \implies (bm si1, bm si2) ∈ same_outs clear)
6 INDEX

dominates ........................................ 5
fef003 ........................................... 4
glb ................................................. 5
lattice_bottom ................................. 5
lattice_top ...................................... 5
lub ............................................... 5
same_ins ........................................ 6
same_outs ...................................... 6
secure .......................................... 7