Project: DRA FRONT END FILTER PROJECT

Title: Phase 3 Technical Overview and Final Report

Ref: DS/FMU/FEF/046 *Issue: Revision* : *2.1* *Date:* 5 June 2016

Status: Approved *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 gives an overview of the formal work carried out under the phase 3 of the FEF project and serves as the final report on that work. It also discusses the relationship between earlier work and the phase 3 work and suggests some possible directions for future research. The work described was carried out under 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/FEF/003. *Formal Security Policy.* G.M. Prout, ICL Secure Systems, WIN01.

[2] DS/FMU/FEF/032. *Table Computations for SWORD.* R.D. Arthan, ICL Secure Systems, WIN01.

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

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

[5] DS/FMU/FEF/041. *Briefing for CLEF.* R.B. Jones and R. Stokes, ICL Secure Systems, WIN01.

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

[7] DS/FMU/FEF/043. *The Labelling Property for SWORD.* R.B. Jones, ICL Secure Systems, WIN01.

[8] DS/FMU/FEF/044. *Proofs About Labelling.* R.B. Jones, ICL Secure Systems, WIN01.

[9] DS/FMU/FEF/045. *Phase 3 Theory Listings.* R.B. Jones, ICL Secure Systems, WIN01.

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

0.3 Changes History

Issue 1.1 (14 March 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

This document gives an overview of the formal work carried out under the phase 3 of the FEF project and serves as the final report on that work. It also discusses the relationship between earlier work and phase 3 work and suggests some possible directions for future research. It constitutes deliverable D19 asked for in the phase 3 proposal [3].

1.2 Introduction

This document is organised into sections as follows:

Section 2 gives an overview of the formal work. It is intended to serve as an introduction to the documents [4, 6, 7].

Section 3 discusses how the earlier single level treatment might be adapted so that the multi-level approach of phase 3 would apply to it.

Section 4 gives some conclusions drawn from the phase 3 work and suggests possible future directions.

Sections 2.2 and 2.3 of this document presuppose familiarity with DRA's original paper on multi-level security policy for SWORD, [10].

2 OVERVIEW OF THE FORMAL TREATMENT

Sections 2.1, 2.2 and 2.3 below discuss the specifications and conjectures made in the documents [4], [6] and [7] respectively.

The proof scripts relating to these three specification documents are collected together in one document [8] which causes each theorem to be saved in the theory containing the specification to which it relates. This is a slight variation on the approach used in phases 1 and 2. A collected theory listing is supplied in [9].

2.1 Multi-level Formal Security Policy

The main topic of [4] is a generic formulation of the no-flows-down policy which can be instantiated to cover both the single level (as in [1]) and the multi-level case (as in [10]). A subsidiary topic is the problem of when extraction of a labelled component from a multi-level object is a secure operation. This latter topic largely serves to give some opportunities for doing proof to validate the generic policy; it also suggests some directions for further theoretical work. The two topics are discussed more fully in sections 2.1.1 and 2.1.2 below.

2.1.1 The Generic Policy

The single level policy of [1] is formulated for "behavioural models", that is to say systems viewed as functions mapping input sequences to output sequences. Since we can treat an input sequence or an output sequence as a single object with an appropriate multi-level structure, it makes little difference in a multi-level formulation whether we model systems as functions mapping sequences to sequences or as functions mapping input objects to output objects. In the following discussion, therefore, we just talk of inputs and outputs.

Informally then, the non-interference style statement of the multi-level no-flows-down policy should read something like:

> *A* multi-level secure system *is a function, b say, such that for any classification c and pair of inputs i_1 and i_2, which appear the same at and below c, the corresponding outputs $b(i_1)$ and $b(i_2)$ also appear the same at and below c.*

As already mentioned, the generality of the definition means that we no longer need to worry about sequences of inputs and outputs. Now, the functional composition of two systems which are multi-level secure in the above sense will again be multi-level secure; thus, one can hope to handle information flows in a state-based system in a uniform way: if there are no illicit flows from the inputs to the state and no illicit flows from the state to the outputs, then one can expect the system to be secure (and could expect a general proof of this fact, given an appropriate rigorous formulation).

Here then, the only thing left to explain is what it means to be the "same at and below classification c". The first goal of [4] is, therefore, a general formulation of this notion. The treatment is influenced by an informal notion of "classified information space" (a name which is not used in [4] because, as suggested below, it is not yet clear what the "right" formalisation is). Very informally, a classified information space is to be a set, X say, endowed with the right additional structure to serve as the

domain or range of a multi-level secure system in the above sense. That is to say, X must come supplied with a way of determining when two elements x_1 and x_2 appear the same at and below a give classification c. One approach to capturing such a property would be to say:

> *A* classified information space, *X, is a set given as the product, $\prod_{c \in Class} V$, where V is some set of values. I.e., the elements of X are functions, x say, on the set Class; the value x_c of such an element at a class c represents the information in x which is classified at c. Two elements x and y are then defined to be the* same at and below classification *c iff. $x_d = y_d$ for any classification d dominated by c.*

There are however problems with this; in particular:

1. In practice, the domains and ranges of systems are not usually presented this way: e.g., the single-level policy is couched in terms of an equivalence relation on the input and output sequences parameterised by the classification (*same_ins* and *same_outs*). Indeed, one might feel that the definition suggested poly-instantiation too strongly to be natural for a system such as SWORD.
2. The definition does not allow for invariants on the inputs. Now, in some ways this is a good thing, because the non-interference style formulation of the no-flows-down policy may well fail to capture the right intuitions given ill-chosen invariants. E.g., assume that inputs are constrained so that there are two components at disjoint classifications, c and d say, which must have the same value (so the invariant is $x_c = x_d$); while it is far from clear what the intention behind such an invariant would be, it certainly prevents the non-interference from properly constraining the potential flows between information classified at c and information classified at d. However, actual systems such as SWORD may have benign invariants of various sorts (e.g. relating to the structure of a database table) and so a definition general enough to allow invariants is required).

As a result of the above considerations, [4] formulates the multi-level policy on the assumption that the domain and range of a system are given the structure of what are called in [4] "indexed equivalence relations". This notion is informally defined as follows:

> *An* indexed equivalence relation *is a family s of equivalence relations on a set X indexed by sets of classifications in such a way that if $A \subseteq B$ are two sets of classes, the equivalence relations s_B associated with B is finer than that associated with A.*

The idea is that the the indexed equivalence relation will be chosen so that, for any set A of classes, two values x and y are related by s_A iff. x and y appear to be the same at all classifications in A. This notion of indexed equivalence relation certainly addresses the problems mentioned above. It turns out to be straightforward to prove that the resulting generic policy generalises the single level ones and the formulation allows for arbitrary invariants to be imposed. In fact, the definition is, perhaps, a little too liberal in that it permits invariants of the malign sort discussed under item 2 above. It is an open question whether there is any natural condition that one can impose which ensures that invariants are benign (see the end of section 2.1.2 for more discussion of this point). [4] therefore defines a generic version of the multi-level security policy using a formulation of the structure required on inputs and outputs based on this notion of indexed equivalence relation. It is conjectured in [4] and proved in [8] that the generic definition does indeed embrace the single level policy of [1] as a special case (see *conj_040_1* in [4]).

2.1.2 Component Extraction

Section 3 of [4] is concerned with exploring one approach to the result labelling property which was initially considered. However, further work on the problem suggested that it would be more useful in the short term to try an approach which exploited more of the specific details of [10]. Thus apart from any intrinsic interest it may have, section 3 of [4] mainly serves to validate some of the definitions in the earlier parts of the document.

The idea in section 3 of [4] is to consider the question: when is the operation of extracting a labelled component from an object a secure operation? Here, by labelled component, we mean a value extracted from the object together with a label giving the classification of that value. More formally, extraction of a labelled component is modelled by two functions V and C say. For an object x, $V(x)$ is the value at x and $C(x)$ is the classification label.

Now, given an arbitrary function, f say on a set, X endowed with an indexed equivalence relation s and given an element x one may consider the set of classes at or above which information has flowed into the result when $f(x)$ is computed. This notion permits a non-interference style formulation, given as the definition of a constant *Influenced* in [4], which is somewhat reminiscent of the definition of *RESULT_INFLUENCES* in [10]. Given s, x and f, *Influenced* $s\ x\ f$ is the set of classes at or above which information has flowed into $f(x)$. Loosely speaking, one would hope that a pair of functions, V and C, would gived a secure means of extracting a labelled component if for each x, *Influenced* $s\ x\ V$ was a set of classes dominated by $C(x)$. That is to say extraction of a labelled component is secure if the label dominates every class at or above which information may have flowed into the extracted value.

It turns out that a little bit of extra machinery is required to make these ideas work properly: the labelling function C is itself a possible source of information flow and so must itself be classified; for simplicity, [4] takes the clearance of the client extracting the component as given and uses that to classify C. With this extra twist, it turns out that a condition like the one mentioned in the previous paragraph is necessary, but not sufficient. The necessity is conjectured in [4] and proved in [8] (see *conj_040_2* in [4]). The fact that the condition is not sufficient is shown informally by an example in [4]. The lack of sufficiency is closely connected with the issue of invariants which undermine the non-interference style formulation of no-flows-down mentioned in section 2.1.1 above. It seems likely that if there are natural criteria under which the condition would be sufficient, then those criteria would be precisely what distingueishes benign invariants from malign ones. However, it is by no means certain whether natural criteria of this sort can be formulated, and more work would need to be done to explore this question. In a specific application, if there were any doubt, one would have to check in some way that the particular indexed equivalence relations used allowed enough freedom for the formulation of the policy to mean what is intended.

2.2 Multi-level Architectural Model

The document [6] gives a fairly straightforward transcription into HOL of the key ideas of [10, section 3]. Certain adjustments have been made in the transcription for the following main reasons:

1. The result labelling property requires us to deal with outputs from SWORD which have not been sanitised (so that we can make assertions about information flow into parts of the outputs of a query which the requesting client is not cleared to see).

2. The formulation of the policy is intended to exploit the generic treatment of such things in [4].

Item 1 here is accomodated in [6] by giving an explicit architectural construction of the system using a construction function *SWORD_construction* which works by filtering the outputs of a non-sanitising machine.

Item 2 is reflected in the way the policy itself is formulated and in the rephrasing of the relevant parts of [10] to fall under the concept of an indexed equivalence relation used in [4].

A more detailed discussion of the relationship between the HOL transcription and the Z treatment in [10] is given in [6] itself, particularly section 1.3.

2.3 The Labelling Property for SWORD

The document [7] is concerned with the result labelling property of [10, section 4]. The labelling property seems to be a rather tricky notion to formalise in general; the approach taken in [7] goes about as far as one usefully can without detailed consideration of the query language semantics. Given a suitably "factored" account of the full semantics for SSQL, the treatment could be instantiated to allow for all the details (although, unless done with care, this could turn out to be a complex task).

In order to validate the ideas, various proofs have been carried out concerning the specifications in [7], including two proofs that the property *label_secure* is satisfiable (i.e., consistent in the sense that their are objects which satisfy it). The proof scripts are given in [8], which also contains the definitions of the two objects which witness the satisfiability: one corresponding to a system which is completely trivial in that its output never has any information content; the other a little less trivial in that its output has the same information content as its input..

The basic idea behind to [7] is to assume that the process of evaluating a (select) query has been factored into four stages. The stages correspond to the four levels of the tree which appears as the result of a query as discussed in [10, section 2.3]. The intuition is of a process which proceeds in four stages. After stage i, $(i = 0, 1, 2, 3)$, levels $0 \ldots i - 1$ of the tree have been completely determined, and level i is thought of as containing an encoding of how the actual values at that level are to be computed. Since there are only four levels, after stage 3 the value of the result must be completely determined.

The approach taken to the result labelling property assumes that a factorisation of the query evaluation process is given. For a given query and input sequence, we may use the factorisation to "pre-evaluate" the first i stages of query evaluation and then assert that the labels are correct at level i if they give a bound on the classification of the information they label (with the information content of the preceding stages effectively factored out).

For example, assume that database contains the table:

Tab

Key		Cargo		Transport	
1	[U]	"Slings"	[S]	"Wagon"	[U]
2	[U]	"Arrows"	[S]	"Wagon"	[U]
3	[U]	"Torpedoes"	[S]	"Ship"	[U]

Consider the query:

SSQL Example

```
SELECT Transport FROM Tab WHERE Cargo = "Torpedoes"
```

Clearly, the secret information accessed by the selection criteria here has been revealed to a client which issues this query and sees the result:

"Ship"	[U]

In the factorisation of the query evaluation, it is intended that by the third stage everything about the result will have been determined except for the actual values in the fields of the rows to be returned. Thus the secret information revealed by the selection criteria for each row has been computed, and no longer contributes to the fourth stage of evaluation. Thus the fourth stage might, perhaps, act like evaluation of the query:

SSQL Example

```
SELECT Transport FROM Tab WHERE Key = 3
```

Information flow analysis of this query shows that it is acceptable to label the result field as unclassified.

Two points should be emphasised however:

1. The information flow analysis on the factored query evaluation does not, of itself, imply that it is secure for a client to downgrade a field which is labelled at a classification below that of the query.
2. The factorisation must be constructed in such a way that the information flow analysis correctly captures intuitions about the labels.

To address the first point one would really need to know more about the client in question and its role in the overall system. It was not felt to be within the scope of the phase 3 work to attempt to identify what clients might attempt to do with a result from the database.

As an example of the second point, consider the query:

SSQL Example

```
SELECT Cargo & Transport FROM Tab WHERE Key = 3
```

If the first three stages of evaluation go too far, then the fourth stage might act like evaluation of the query:

SSQL Example

```
SELECT "Torpedoes" & Transport FROM Tab WHERE Key = 3
```

Here, the information flow analysis would permit the labelling:

"TorpedoesShip"	[U]

which is intuitively incorrect, since information from a field classified secret appears in the result. More detailed consideration of the structure of the query language would presumably enable one to state formally how far each stage may go.

3 APPLYING THE RESULTS

This section considers the application of the results of phase 3 to the earlier single level treatment as used in phase 1 or phase 2. For definiteness, and because it was the phase in which detailed reasoning about the query language evaluation was first required, we will restrict attention here to phase 2 (although most of what is said is equally applicable to the phase 1 work on the query language semantics).

A number of architectural and policy issues would need to be considered before applying the phase 3 work to the earlier treatments. The phase 3 statement of the security policy is stated in [6] using a single level view of the output of the system. This reflects the fact that SWORD does not return unsanitised data to the client. In the Z parts of [10], *MACHINEs* are modelled as if their outputs were unsanitised multi-level objects. In [6, 7], where unsanitised multi-level output objects are needed (e.g., to specify the labelling property), they are the outputs of an internal component of the architecture. These architectural differences would need to be reconciled and the relationship between the result labelling property and the policy itself considered.

In [10], it is suggested that the result labelling property is to be exploited by certain clients to perform security-critical functions such as downgrades. It might be best to take the expected behaviour of such clients into account with a view to defining an overall system security policy. The result labelling property might then appear as a critical property of the SWORD subsystem used to prove the overall security. Such an approach would help to validate the result labelling property against a policy which would depend less on internal details of the implementation and the query language semantics.

As regards the supplying an appropriate factorisation of the query evaluation, the treatment of the select query in [2] is suggestive. The function *AllTuples* there which models the select query, is already organised as the composite of the composite of four subsidiary operations, *Join, Where, Group* and *Project*. The last of these operations corresponds quite closely with what one would expect to happen at the fourth stage of query evaluation as discussed in section labelling above.

4 CONCLUSIONS AND FUTURE DIRECTIONS

A formulation in HOL of a multi-level treatment of the security policy for SWORD along the lines suggested in [10] has been investigated. Several approaches to the result labelling property have been considered and an approach using a factorisation of the query evaluation process has been selected as the most promising. In order to validate the specification work several non-trivial conjectures have been stated and proved.

The multi-level policy has been formulated in a generic way; it is parameterised by the information required to view the inputs and outputs of a system as multi-level objects. As with any non-interference policy, care must be taken to ensure that invariants on the inputs and outputs do not subvert the means for prohibiting information flows. It would be of theoretical interest and practical value to have a better criteria for recognising such malign invariants; the framework of [4] provides a simple but general context in which to investigate this.

The work on result labelling would probably be best progressed by attempting to relate it to some possible scenarios of use for SWORD. Given more knowledge of the circumstances under which clients are likely to use labels for downgrades, one might try to validate the result labelling property against an overall security policy for the clients and SWORD together. Once this was done, it would then

be appropriate to reconcile the architectural mismatches between the [5] and the phase 2 treatment and then to adapt the existing single-level treatment in the light of the multi-level policy and the result labelling property.