Proof of Non-Interference Unwinding Theorem

William D. Young
University of Texas at Austin

Assume a deterministic system with a set A of agents (subjects), a set S of states, and a set $\mathcal{I}$ of instructions. Transitions in the system are defined in terms of the function $step : \mathcal{I} \times \mathcal{S} \rightarrow \mathcal{S}$. Another function, $agent : \mathcal{I} \rightarrow \mathcal{A}$, associates with each instruction the agent executing that instruction.

The system security policy is defined in terms of a boolean-valued binary "interference" relation: $\mapsto: A \times A$. The intended interpretation of this is: agent a is allowed to communicate[1] with b iff $a \mapsto b$. Finally, define another set S_A that is those collections of system components visible to a given agent. The function $view : A \times S \rightarrow S_A$ is intended to associate with an agent and state pair, the portion of state visible to that agent.

The following is the definition of non-interference security for such a system:

Definition 1: The system is non-interference secure iff:

$$\forall a \in A, \forall S_0 \in S, \forall I \in \mathcal{I}^* : view(a, run(I, S_0)) = view(a, run(purge(I, a), S_0))$$

The definitions of functions *run* and *purge* are provided in the appendix.

The following two definitions constitute sufficient *unwinding conditions* for non-interference security in such a system.

Definition 2: The system *locally respects* the interference relation $\mapsto$ iff:

$$\forall a \in A, \forall s \in S, \forall i \in \mathcal{I} : [agent(i) \not\mapsto a] \Rightarrow [view(a, step(i, s)) = view(a, s)]$$

Definition 3: The system is *step consistent* iff:

$$\forall a \in A, \forall s_1, s_2 \in S, \forall i \in \mathcal{I} :$$
$$[view(a, s_1) = view(a, s_2)] \Rightarrow [view(a, step(i, s_1)) = view(a, step(i, s_2))]$$

[1] More precisely, information is allowed to flow from the domain of a to the domain of b. Whether that flow is by action of a, action of b, or both is not specified.

What it means for these to be unwinding conditions is expressed in the following theorem.

Theorem: A deterministic system with transitive interference relation $\mapsto$ and in which locally respects and step consistency both hold is non-interference secure.

Proof: Assume an arbitrary agent a, initial state S_0, and instruction sequence I. We need to show that:

$$view(a, run(I, S_0)) = view(a, run(purge(I, a), S_0)).$$

The proof is by structural induction on the instruction sequence I.

Base case: ($I = nil$). By the definitions of *run* and *purge*, both sides reduce to $view(a, S_0)$.

Induction step: ($I = I' \circ i$). Assume $agent(i) = b$. By the induction hypothesis we assume that,

$$view(a, run(I', S_0)) = view(a, run(purge(I', a), S_0)).$$

Working on the left hand side of our theorem:

$$view(a, run(I, S_0)) = view(a, run(I' \circ i, S_0)) = view(a, step(i, run(I', S_0))$$

by the definition of *run*.

At this point, we need to consider two possibilities: either b is allowed to interfere with a or is not.

Case 1: ($b \not\mapsto a$) In this case, by locally respects, the final form above is equal to:

$$view(a, run(I', S_0)).$$

Working on the right hand side of our theorem,

$$view(a, run(purge(I, a), S_0)) = view(a, run(purge(I' \circ i, a), S_0)).$$

But since $b \not\mapsto a$, by the definition of *purge* this becomes:

$$view(a, run(purge(I', a), S_0))$$

which is then equal to the left hand side by the induction hypothesis.

Case 2: ($b \mapsto a$) Again, working on the right hand side of our theorem:

$$view(a, run(purge(I, a), S_0)) = view(a, run(purge(I' \circ i, a), S_0)).$$

In this case since $b \mapsto a$, by the definition of *purge*, this becomes:

$$view\,(a, run\,(purge\,(I', a) \circ i, S_0)).$$

Then, by the definition of *run*, this is equal to:

$$view\,(a, step\,(i, run\,(purge\,(I', a), S_0))).$$

But by the induction hypothesis, we know that:

$$view\,(a, run\,(I', S_0)) = view\,(a, run\,(purge\,(I', a), S_0)).$$

According to step consistency, if two states are view-identical for any agent a, then executing the same instruction in both will result in states that are view-identical for a. Consequently,

$$view\,(a, step\,(i, run\,(I', S_0))) = view\,(a, step\,(i, run\,(purge\,(I', a), S_0)))$$

which proves our theorem.

Appendix

Below are the definitions of the functions *run* and *purge*.

Definition A1: The function $run : \mathcal{I}^* \times \mathcal{S} \to \mathcal{S}$ maps an instruction sequence and a state to a state as follows.

$$\begin{array}{rcl} run\,(nil,\, a) & = & a \\ run\,(l \circ i,\, a) & = & step\,(i,\, run\,(l,\, a)) \end{array}$$

Here *nil* denotes the empty sequence, and $l \circ i$ denotes the concatenation of element i to the right end of sequence l.

Definition A2: The function $purge : \mathcal{I}^* \times \mathcal{A} \to \mathcal{I}^*$ maps a sequence of instructions and an agent to a sequence of instructions as follows.

$$\begin{array}{rcl} purge\,(nil,\, a) & = & nil \\ purge\,(l \circ i,\, a) & = & \begin{cases} purge\,(l,\, a) \circ i, & \text{if } agent\,(i) \mapsto a \\ purge\,(l,\, a), & \text{otherwise} \end{cases} \end{array}$$