

CS4605/Lab 3

George W. Dinolt

August 3, 2004

1 introduction

The goal of this lab is to introduce the framework for formally describing and analyzing various security models. As we have mentioned in class, we will be looking at a hierarchical approach for organizing and viewing the specifications. Figure 1 shows the general arrangement of the specifications.

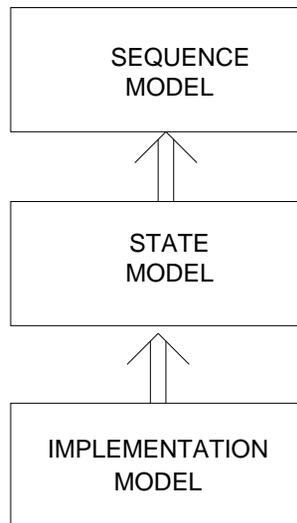


Figure 1: The layering of the Models

The top level is what is sometimes called the *Basic Security Theorem*. Its main purpose is to show that a sequence which is built in the proper way will be secure. The second level is an abstraction of the *State Machine* that is assumed in the top

level. The third level is a model of the implementation. This might represent the *Formal Top Level Specification* in various descriptions.

In this Lab we will concentrate on the top level, the **Sequence Model**.

2 The Specification of the Top Layer

The specification is shown in Figure 2 You can download a copy of the theory

```
seq_theory[State: TYPE, Inputs: TYPE, seq: se-
quence[State], st?: [State → bool], transform:
    [Inputs, State → State]]: THEORY
BEGIN

ASSUMING
  seq_0_secure: ASSUMPTION st?(nth(seq, 0))

  transition_state_secure: ASSUMPTION
    ∀ (st: State), (x: Inputs): st?(st) ⇒ st?(transform(x, st))

  seq_transform: ASSUMPTION
    ∀ (n: nat):
      ∃ (x: Inputs): nth(seq, n + 1) = trans-
form(x, nth(seq, n))
ENDASSUMING

seq_is_secure: THEOREM every(st?)(seq)

END seq_theory
```

Figure 2: The specification of the top level Sequence Theory

from the web site here. ¹

The `seq_theory` specification provides an environment for the main theorem. We assume that we have a state machine with states from the State **TYPE**.

¹This is a *hot-link* and points to the file
http://www.nps.navy.mil/cs/Dinolt/Courses/AY2004/Winter/CS4605/Labs/lab3/seq_theory.pvs.
 You can also download a copy from `/disk1/cisr/pvs-examples` on proof.

This is defined in the parameters of the theory . The *alphabet* of the state machine is the set of `Inputs`. An execution of the state machine is one of the elements of the `seq` TYPE.

We also define two functions on states. The first is a test to determine whether the state is *secure*.² The function is written as *st?* where the ? is used to indicate that it returns either `true` or `false`.

The second function, *transform* is the function that shows how to transition from one state to the next. The idea is that the new state depends on the previous state and the input that was received.

The main section of the specification is bounded by `ASSUMING` and `ENDASSUMING`. These are the assumptions that this specification makes about the properties of the parameters of the specification. The idea is that if the sequence `seq` satisfies these assumptions then every term of `seq` will be secure, i.e. every term of the sequence is secure or,³

$$\forall n : st?(nth(seq, n)) = true.$$

The **THEOREM** is really an induction on the elements of the sequence. So we have to show the base case and how to proceed from one element of the sequence to the next. Below we describe the role of each of the assumptions.

seq_0_secure

This is the base case. It says that the 0^{th} element of the sequence is secure.

seq_transform

This function shows that we can explicitly transform one state to another using an input.

transition_state_secure

This function shows that if one starts in a secure state and does a *transform* as described above, one ends up in a secure state.

²At this level of the specification, we don't know what this means. For any particular system we will have to instantiate it.

³The *nth* function returns element *n* of the *seq*.

3 What you need to do for the Lab

You need to download the specification onto the system you will be working on. You should provide me a proof of the *seq_is_secure* theorem. You will need to use the commands

- *expand*
- *induct*
- *use*
- *skolem!* (twice?) — Note that *skolem!* instantiates “for all’s” below the line and “exists” above the line.
- *replace*

as well as those you have used before. You will need to look these up to see how they can be used. You should hand-in the show-proof results.

Good Luck