Using SPARK-Ada to Model and Verify a MILS Message Router*

Bryan Rossebo, Paul Oman, Jim Alves-Foss, Ryan Blue, and Paul Jaszkowiak
Center for Secure and Dependable Systems
University of Idaho
Moscow, ID 83844-1008

Abstract

The concept of information classification is used by all nations to control information distribution and access. In the United States this is referred to as Multiple Levels of Security (MLS), which includes designations for unclassified, confidential, secret, and top secret information. The U.S. Department of Defense has traditionally implemented MLS separation via discrete physical devices, but with the transformation of military doctrine to net-centric warfare, the desire to have a single device capable of Multiple Independent Levels of Security (MILS) emerged. In this paper we present a formal model of a MILS message router using SPARK-ADA. The model is presented as a case study for the design and verification of high assurance computing systems in the presence of an underlying separation kernel. We utilized the correctness-by-design approach to secure system development and discuss the limitations of that approach for the type of system we model.

1. The need for certifiably secure systems

One of the largest problems facing the field of computer science is that of computer and network security. With the increased connectivity of Information Technology (IT) systems and process control systems, security is needed to defend against malicious persons intent on abusing or attacking network resources. This is especially true for unbounded networks like the Global Information Grid (GIG) [1].

Every year, billions of dollars are lost due to cyber intrusions and computer viruses that threaten corporate and government systems. The "ILOVEYOU" virus alone

Report Documentation Page

Form Approved
OMB No. 0704-0188

greatly simplify the work of building and certifying MILS-based EAL7 systems for critical uses.

3. A SPARK-Ada MMR model

The main goal of the research described in this paper is to model the MMR and then verify the model's correct operation via formal methods. One of the principle objectives of the MILS initiative is the creation of EAL7-certified components. EAL7 requires the entire system to be mathematically proven using formal methods [4]. Specifically, we must create a high-level MMR design that is (a) proven in a formal modeling language, and (b) traceable to the code implementation [5]. SPARK-Ada is a formal methods tool that facilitates both criteria; specifically, it incorporates formal Hoare logic operations with executable Ada code

fact operate correctly to facilitate secure messaging, and
(b) SPARK-Ada is a viable proobis a k6v3.5(oe1.2(nrific)7.2(btionto)-.5(o)l)6(fa)-.5(o)4.2(o)]TJ0 -1.1493 TD0.Tc0.23189Tw[(exc

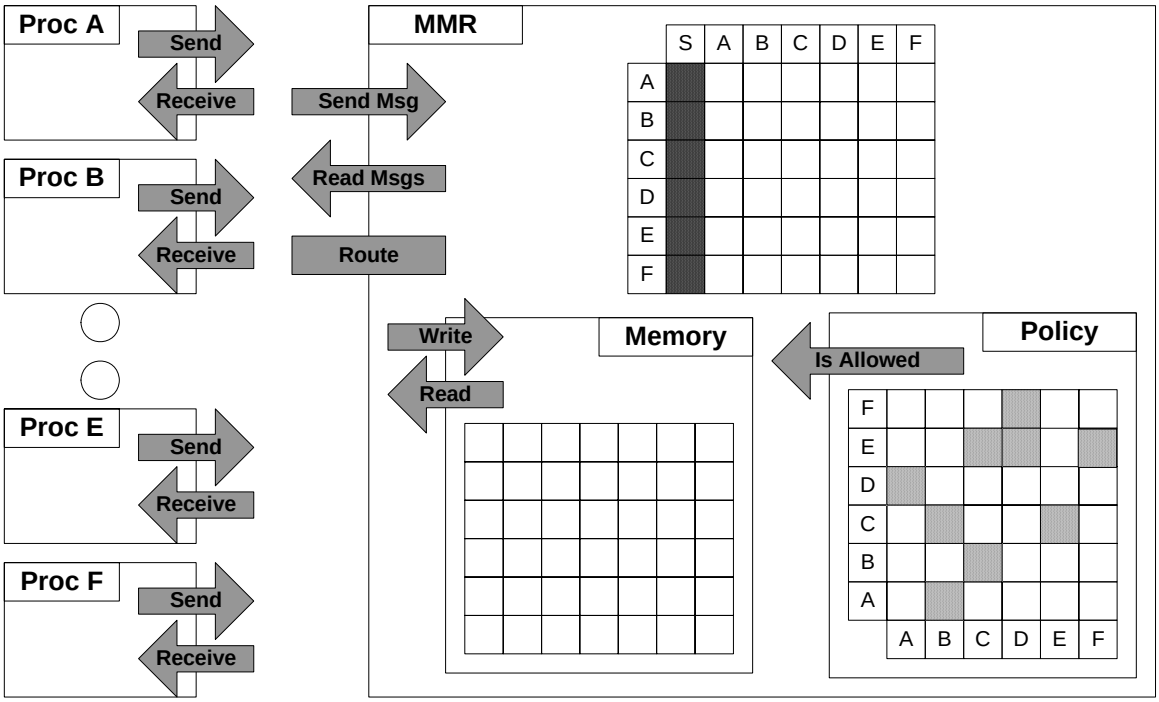

Figure 4. MMR interactions with processes A-F

```
--# inherit Lbl_t, Msg_t;
package Memory
  --# own Mem_Space : Mem_Space_T;
  --# initializes Mem_Space;
  is
  type Mem_Space_T is array
     ( Lbl_t.Pointer ) of Msg_t.Msg;
  procedure Write(
     M: in Msg_t.Msg;
     S: in Lbl_t.Pointer );
  --# global in out Mem_Space;
  --# derives Mem_Space from *,
  --#                  M,
  --#                  S;
  --# post Mem_Space = Mem_Space~[ S => M ];
  procedure Read(
     M: out Msg_t.Msg;
     S: in  Lbl_t.Pointer );
  --# global in Mem_Space;
  --# derives M from Mem_Space,
  --#             S;
  --# post M = Mem_Space( S );
end Memory;
```

(a) *Memory* package specification

```
package body Memory is
  Mem_Space: Mem_Space_T;
  procedure Write(
     M: in Msg_t.Msg;
     S: in Lbl_t.Pointer ) is
  begin
    Mem_Space( S ) := M;
  end Write;
  procedure Read(
     M: out Msg_t.Msg;
     S: in  Lbl_t.Pointer ) is
  begin
    M := Mem_Space( S );
  end Read;
begin
  Mem_Space := Mem_Space_T'(
   Lbl_t.Pointer => Msg_t.Def_Msg );
end Memory;
```

(b) *Memory* package body

Figure 5. SPARK-Ada code for *Memory* package

6. The *Main* package is just a wrapper program that executes the *System* package indefinitely.

Fig. 4 depicts the relationship between the *MMR* package, with its subordinate *Memory* and *Policy* packages, and the simulated processes denoted A through F. The *Policy* package contains an adjacency matrix representing the security policy diagraph shown in Fig. 3. The figure shows how the *MMR* is designed to interact in the model, where the partition in each row is allowed to talk to the partition in the column if the cell is shaded. The *MMR* has only three publicly available procedures: *Send_Msg*, *Read_Msgs*, and *Route*. There are two internal packages within the *MMR*: *Memory* that has two procedures, *Write* and *Read*, and *Policy* that has one

function, *Is_Allowed*. There is also an internal table within the *MMR*

SPADE Proof Checker in manually guide mode. The proof checker program takes in <name>.vcg or <name>.siv files containing unverified conditions and outputs the manually guided verifications into <name>.plg files. In a similar manner, the review team can create verifications in a <name>.prv file containing verification conditions that have been manually verified by a review committee.

When all verification conditions have been proven, either automatically or manually, the Proof Obligation Summarizer (POGS) checks for the existence of all files

- Tying the executable code to the formal proof assertions (*a la* SPARK-Ada) enables a more rigorous proof model than can be attained through non-executable formal methods proof environments in which we have worked (e.g., ACL2).
- The lack of commercial grade on-call assistance on the use and nuances of the SPARK-Ada verification toolset was a significant hindrance to our task.