AN EXPERIENCE MODELING CRITICAL REQUIREMENTS

Charles N. Payne, Jr. Andrew P. Moore, and David M. Mihelcic
Center for High Assurance Computing Systems
Naval Research Laboratory
Washington, D.C. 20375-5337

Abstract

Previous work at NRL demonstrated the benefits of a security modeling approach for building high assurance systems for particular application domains. This paper[1] introduces an application domain called *selective bypass* that is prominent in certain network security solutions. We present a parameterized modeling framework for the domain and then instantiate a confidentiality model for a particular application, called the *External COMSEC Adaptor (ECA)*, within the framework. We conclude with lessons we learned from modeling, implementing and verifying the ECA. Our experience supports the use of the application-based security modeling approach for high assurance systems.

Introduction

Conventional security modeling approaches (c.f., [1]) usually express a system's confidentiality requirements independently of the application that the system must support. Previous work at NRL [10, 11] argued that users understand a system's *critical requirements* (i.e., the assertions that the system must enforce and the assumptions about the system's environment) better if the model is expressed in terms of the application. Improved user understanding can reduce unintentional compromises in system security, making the application-specific model a better foundation for building operationally secure systems. The proponents of the approach applied it to the security requirements for a family of military message systems (MMS) [10]. Unfortunately, a trustworthy implementation was never built, leaving unresolved questions about the practical utility of the approach.

Our motivation is to develop a series of increasingly sophisticated systems for a particular application domain. In [14], McLean demonstrated a modeling framework that includes parameters for constructing models whose critical requirements range from intuitively weak to intuitively strong. The parameters' values reflect the degree of risk that the system's environment will accept. In this paper, we describe an application domain called *selective bypass*, and we present a parameterized framework for constructing application-specific models in that domain. Then we identify a particular application within the domain called the External COMSEC Adaptor (ECA) and instantiate a model for it.[2] Finally, based on our experience implementing the ECA and verifying that the implementation satisfies the model assertions, we present some lessons about the modeling activity and its impact on the verification effort.

Our experience with the ECA supports the application-based modeling framework. We believe that the framework for the Selective Bypass Domain provides a good starting point for developers of applications in that domain. We expect that the lessons presented will be helpful for developers that wish to use the application-based modeling framework for building trustworthy systems in this and other application domains.

The Selective Bypass Domain

The Selective Bypass Domain contains applications responsible for providing cryptographic protection of information based on the rules that determine the sensitivity of the information. We refer to a particular member of the domain as a *Selective Bypass Device* (SBD). The primary use of an SBD is to provide some degree of security for the network in which it is embedded.

The SBD must separate two security levels of information: sensitive and non-sensitive. Sensitive information is made non-sensitive through its encryption using a secret key. The determination that information is sensitive—and thus in need of encryption—is a policy decision of the application system that contains the SBD. The decision is encoded as a set of rules in the SBD. These rules may, due to the complexity of the

[1] Published in Proc. COMPASS 94, Gaithersburg, MD, IEEE Press, IEEE Cat. 94CH3415-7, ISBN 0-7803-1855-2, June 1994, pp. 245–256.

[2] An earlier version of this model can be found in [24].

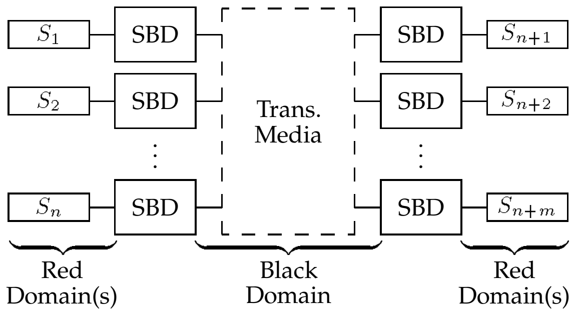

Figure 1: The Network

task, only coarsely determine the actual sensitivity of information (e.g., a "dirty word" check).

The Environment

The SBD resides in a network that supports secure communications. The network consists of subscribers, SBDs and transmission media. Anyone that can access the network is considered a user. Although legal access is primarily through a subscriber, other legal access might include the secure network management function and the cryptographic key loading function. Subscribers range from simple PCs to multilevel-secure systems and are accredited to process messages up to a particular classification level. Figure 1 illustrates a very important characteristic of the network: every subscriber, i.e., S_1, S_2, ..., S_{n+m}, is connected to a unique SBD, and all communications between subscribers must pass through the subscribers' SBDs.

The network enforces a simple security policy of data confidentiality: users shall not receive sensitive information for which they are not authorized. Confidentiality is achieved by partitioning the network into a domain for processing possibly sensitive plaintext data, called the Red Domain, and a domain for processing nonsensitive and encrypted sensitive data, called the Black Domain. Since each subscriber can be accredited to a different sensitivity level, there may be multiple Red Domains in the network. Users residing in a Red Domain are trusted to protect the information they process to a degree appropriate for the security classification of the data. Users residing in the Black Domain are assumed to be malicious and should not have access to the plaintext sensitive data. A communications plan determines which subscribers (and consequently, users) may communicate, and the network relies on cryptographic key distribution to implement the plan. The SBD contains the cryptographic function.

The SBD Security Policy

Figure 1 illustrates the SBD's role for partitioning the Red Domain and the Black Domain. The sensitive information of a message must be made non-sensitive by the SBD before the message may be transmitted over the transmission media. The SBD must enforce the following security policy:

> **Restricted Red-To-Black Flow.** Sensitive information shall not be transmitted through the SBD to the Black Domain.

Restricted Red-To-Black Flow is best characterized as a very simple information flow policy. Unfortunately, the cost of constructing a rigorous assurance argument that the SBD satisfies this policy outweighs the benefits derived from the effort, because the existence of any bypassed data imposes a risk on the use of the SBD. However, many operational environments are willing to accept this risk in order to improve the function and connectivity of their information systems, so we consider **Restricted Red-To-Black Flow** to be an ideal—rather than a strict—requirement.

To reduce the cost of the assurance argument, we identify a set of safety properties that together support **Restricted Red-To-Black Flow**. These safety properties, stated as assertions and assumptions, are preserved using conventional refinement techniques and are simpler to interpret for a particular environment. We use **Restricted Red-To-Black Flow** to assess the sufficiency of the assertions and assumptions and to identify covert channels that might arise during refinement.

Obviously cryptography will play a major role in supporting **Restricted Red-To-Black Flow**; however, messages must be routed properly, and the routing functions are part of the transmission media in the Black Domain. Therefore, a message's routing information, or message header, must be transmitted as plaintext. The SBD must ensure that the routing information is not sensitive. The SBD must also ensure that sensitive information is not encoded in less overt ways in the message stream transmitted to the Black Domain (e.g., rate/length modulation).

Because of the broad range of security provided by members of the Selective Bypass Domain, we constructed a parameterized modeling framework for the domain. This permits us to delay the definition of security-critical parameters until details of the operating environment are known and until a realistic risk assessment can be made. The parameters allow us to construct SBDs that lie between the following extremes:

Most Secure: SBDs that ensure that no sensitive data is encoded in the message stream transmitted to the Black Domain (this implies that all data is encrypted),

Least Secure: SBDs that do not protect the message stream transmitted to the Black Domain (this implies that all data bypasses the cryptographic function).

Assuming that the SBD encryption function properly transforms information from sensitive to non-sensitive, a Most Secure SBD satisfies **Restricted Red-To-Black Flow**. Achieving security in this way is most closely related to Sutherland's definition of multi-level security based on non-deducibility [25]. Although the low level user may be able to read the encrypted data, he will not be able to deduce its higher level meaning.

The SBD Modeling Framework

In this section, we describe briefly the SBD's external interface and user-visible behavior. We identify the parameters upon which the framework is based and express assertions for the SBD in terms of those parameters. We will also identify the assumptions that underlie the assertions.

User's View

The SBD is an embedded system. It has no direct human users, so a "user's" view of its operation must be interpreted for the devices to which it connects. Figure 2 illustrates this view for the devices that connect directly to the message interfaces of the SBD. Devices communicate with the SBD over the message interfaces only. A device in the Red Domain engaged in transmitting and receiving messages communicates only over the message interface in the Red Domain. A device in the Black Domain engaged in transmitting and receiving messages communicates only over the message interface in the Black Domain. Devices communicate with the SBD under an established protocol. Progress of a transmitted message can be relayed to the originating device if the notification does not violate the critical requirements.

In general, the SBD may receive cryptographic keys via its cryptographic interface (illustrated in Figure 2) or over the network from an authorized source. A communications plan defines how to maintain a set of keys, and based on the message, the SBD chooses the proper key from this set. Cryptographic keys may expire, so the SBD must select the most current key for a message.

A message may be partitioned into a crypto data portion that contains sensitive text supplied by the user, and a bypass data portion that contains the transmission protocol information. The bypass data portion can be divided into distinct fields. Each field contains information, such as the source and destination address, needed to route the message and to choose the appropriate cryptographic key. While most messages processed by the SBD originate externally, the SBD can also generate certain control-oriented messages. The SBD uses these internally generated messages to synchronize with the subscriber and transmission media and to report problems in message processing.

In a typical scenario, a message is transmitted from a user (not shown in Figure 2) accessing subscriber S to some remote user. The SBD that is local to S receives the message from S over the message interface in the Red Domain, splits the message into the bypass data and crypto data portions, encrypts the crypto data with the cryptographic function E using the appropriate key k, diverts the bypass data around E, recombines the message and transmits the result over the message interface to the Black Domain. The intended recipient's SBD must reverse this procedure using the corresponding decryption function and key to restore the original message.

The SBD Parameters

The following parameters are used to instantiate the SBD framework for a particular application. The purpose for each will be clarified when we present the assertions. Henceforth, the parameter names appear in SMALL CAPS.

1. KEY DISTRIBUTION AND SELECTION POLICY: determines how to maintain a valid (i.e., current) set of keys and how to select a key for encrypting and decrypting a particular message,

2. FORMAT RESTRICTION: determines the sensitivity of data according to a set of rules,

3. CONTEXT CONSTRAINT: determines the contexts appropriate for transmitting a given message,

4. MESSAGE LENGTH SET: the set of lengths appropriate for a message,

5. BYPASS DATA LENGTH SET: the set of lengths appropriate for bypass data,

6. TRANSMISSION RATE SET: the set of acceptable rates for message transmission,

7. BYPASS RATE CONSTRAINT: determines the maximum limit on the rate of data bypassed around the cryptographic function, and

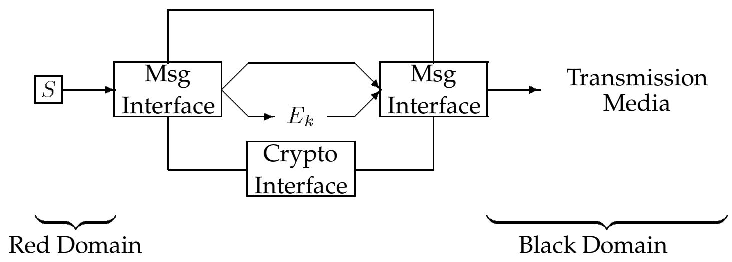

Figure 2: Operation of the SBD

8. PRIORITY MESSAGE-ORDERING SCHEME: assigns a processing priority to messages.

Assumptions

The assertions for the SBD are requirements on its externally-visible, security-relevant behavior. However, this behavior is not generated in a vacuum. The SBD relies heavily on its environment for its correct operation. This dependency on the environment is expressed in terms of assumptions. If we can demonstrate that the SBD satisfies its assertions, and if we can validate the following assumptions, then we can assert that the SBD behaves securely.

A1. **(Physically Secure)** The SBD operates in a physical environment appropriate for the data it processes, i.e., it is physically secure.

A2. **(Valid Parameters)** The parameter values are appropriate for the message set being processed and for the network's security policy.

A3. **(Valid Crypto Algorithm)** The SBD is loaded with a cryptographic algorithm that is appropriate for the message set it processes and for the network's security policy.

One may think of these assumptions as assertions for the SBD's environment. They may be satisfied by a variety of countermeasures, including the use of controlled physical spaces (A1) and through carefully followed administrative procedures (A2 and A3). However, if these assumptions are not validated, a security vulnerability exists for the SBD and its environment [23].

Assertions

Below are the security assertions that the SBD must enforce. These assertions are safety properties that support **Restricted Red-To-Black Flow**.

P1. **(Encryption)** The crypto data of every message transmitted to the Black Domain shall be encrypted according to the KEY DISTRIBUTION AND SELECTION POLICY.

P2. **(Bypass Format)** The bypass data of every message transmitted to the Black Domain shall satisfy its FORMAT RESTRICTION.

P3. **(Message Context)** The context of every message within the sequence transmitted to the Black Domain shall satisfy the CONTEXT CONSTRAINT.

P4. **(Message Length Modulation)** The length of every message transmitted to the Black Domain shall be in the MESSAGE LENGTH SET, e.g., by padding messages as necessary.

P5. **(Bypass Data Length Modulation)** The length of the bypass data of every message transmitted to the Black Domain shall be in the BYPASS DATA LENGTH SET, e.g., by padding bypass data as necessary.

P6. **(Transmission Rate Modulation)** The transmission rate of messages to the Black Domain shall be in the TRANSMISSION RATE SET, e.g., by transmitting dummy messages when no legitimate message is available.

P7. **(Bypass Rate Limitation)** The rate of data bypassed around the cryptographic function shall be limited according to the BYPASS RATE CONSTRAINT.

P8. **(Message Order)** Messages shall be transmitted to the Black Domain according to the PRIORITY MESSAGE-ORDERING SCHEME, and each message shall be transmitted only once.

To show that these safety properties are sufficient to support **Restricted Red-To-Black Flow**, we instantiate a Most Secure SBD and argue informally that it satisfies **Restricted Red-To-Black Flow**.[3] A Most Secure SBD ensures that no sensitive data is encoded in the message stream transmitted to the Black Domain. One response is to state the FORMAT RESTRICTION such that the entire message must be encrypted before it is transmitted to the Black Domain. Since there is no bypass data, assertions P5 and P7 can be satisfied trivially with any non-null instantiation of the BYPASS DATA LENGTH SET and the BYPASS RATE CONSTRAINT. In addition, P3 and P8 were orginally specified to close signaling channels that exist only in the presence of bypass data, so these assertions are irrelevant. Consequently, the only assertions that are significant for a Most Secure SBD are P1, P2, P4 and P6. Given the stringent FORMAT RESTRICTION described above, we must demonstrate that we can instantiate the KEY DISTRIBUTION AND SELECTION POLICY, the MESSAGE LENGTH SET and the TRANSMISSION RATE SET with values that make the four assertions sufficient to satisfy **Restricted Red-To-Black Flow**.

As long as a legitimate KEY DISTRIBUTION AND SELECTION POLICY is chosen (and enforced by the ECA) and as long as all messages are encrypted completely, there are only two ways to signal information: by modulating a message's length or by modulating the rate of message transmission. To address these attacks, we can instantiate the MESSAGE LENGTH SET and the TRANSMISSION RATE SET as singleton sets, i.e., all messages transmitted to the Black Domain must be the same length and all messages are transmitted to the Black Domain at a constant rate. This effectively prevents any signaling to the Black Domain, so a Most Secure SBD satisfies **Restricted Red-To-Black Flow**.

The ECA Model

In this section, we instantiate the SBD model for the ECA application by assigning parameter values that are appropriate for the ECA's environment. We describe the ECA in its environment, present the instantiation and describe the values chosen.

Environmental Constraints on the ECA

The ECA is an SBD that is embedded in a network that must satisfy real-time operational constraints with relatively low bandwidth/high noise communication links. Depending on the values of its parameters, an SBD can provide a variety of security services to a network. However, as we will discuss below, some of these services were neither required nor desired for the ECA.

[3] While we have not done so, we believe this argument can be formalized.

The biggest environmental constraint on the ECA was the decision at the network level to use a particular off-the-shelf cryptographic device in the ECA. While this decision made the ECA easier to implement, it also decreased its functionality. For example, the cryptographic device does not support over-the-network re-key, so the key for processing messages is fixed while the ECA is attached to the network, i.e., the ECA has to be taken off-line to perform the re-key function.

The network's rigid performance requirements precluded any measures to reduce signaling channels through modulation of the message length, bypass data length and message transmission rate. We assigned values to the MESSAGE LENGTH SET, the BYPASS DATA LENGTH SET and the TRANSMISSION RATE SET that would make assertions stated in terms of these parameters (P4, P5 and P6, respectively) irrelevant for the ECA. Similarly, because we knew little about the network's traffic flow—the network itself was still under development—or about its user community, we hesitated to speculate on an appropriate instantiation of the CONTEXT CONSTRAINT, so we assigned it a value that made P3 irrelevant for the ECA also.

As a result, the ECA implements only a core of the SBD's security services. The ECA's network compensates for the missing services by encrypting all messages received from the ECA (i.e., doubly encrypting the crypto data) before they are forwarded to the transmission media. As Figure 3 illustrates, once a message exits the ECA's message interface in the Black Domain, it is transmitted via a somewhat protected routing function to a link encryptor (LE). This subsystem, composed of the ECA, protected routing function and LE, guards against common traffic analysis threats [27].

Assigning Parameter Values for the ECA

The SBD parameter values assigned to the ECA are identified in Table 1. In this section, we expound on each choice and identify any new assumptions that arise as a result. The assumptions identified earlier for the SBD framework also hold for the ECA.

KEY DISTRIBUTION AND SELECTION POLICY. Given the tight development schedule for the ECA, using an off-the-shelf cryptographic device with fixed key and off-line re-keying made sense. However because of these limitations, the following assumptions must be valid in the ECA's environment.

A2.1 Subscribers must be accredited to process data up to the sensitivity level appropriate for the installed key.

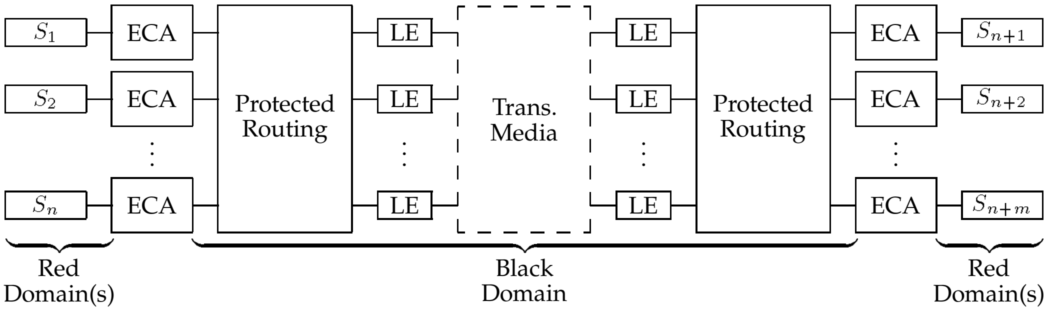

Figure 3: The ECA in the Network

Parameter	**Value**
KEY DISTRIBUTION AND SELECTION POLICY	Fixed key
FORMAT RESTRICTION	Definition of *FormatOK*
CONTEXT CONSTRAINT	None
MESSAGE LENGTH SET	Set of Natural Numbers
BYPASS DATA LENGTH SET	Set of Natural Numbers
TRANSMISSION RATE SET	Set of positive Rational Numbers
BYPASS RATE CONSTRAINT	Definition of *BypassOK*
PRIORITY MESSAGE-ORDERING SCHEME	First in, first out

Table 1: SBD Parameter Instantiation for the ECA

A2.2 The ECA is disconnected from the network during the re-key operation.

FORMAT RESTRICTION. A FORMAT RESTRICTION is defined for each message type, and the demonstration that a message satisfies that restriction is embodied in the following definition:

FormatOK: The value of each field of the bypass data must be within a predetermined range; the length of each field must match a predetermined length for that field; and the overall length of the bypass data, as specified by a field within the bypass data, must equal the sum of the lengths of the fields of the bypass data.

A message is dynamic, and the length and number of any field in the bypass data can depend on the values of previous fields. To limit the memory needed to store complex FORMAT RESTRICTION tables, each field of the bypass data is checked against a range, rather than a set, of values.

BYPASS RATE CONSTRAINT. The ECA relies on two types of checks to limit the flow of bypass data to the Black Domain. An "absolute" check, i.e., restricting the maximum number of bits bypassed per second, is performed in hardware, while an "average" check, i.e., restricting the number of bits bypassed given the time elapsed, is handled redundantly in both hardware and software. After power is initially applied to the ECA and to the other system components, a relatively large number of bits must be bypassed for system initialization. To avoid a time delay (imposed by the average rate restriction), an initial bypass "credit" is provided using a system initialization constant.

BypassOK: The actual rate at which bypass data are transmitted to the Black Domain shall not exceed the allowed rate.

The actual rate is calculated by

$$\frac{B}{T},$$

where B represents the number of bits actually bypassed and $T > 0$ represents the time elapsed. The allowed rate is given by

$$\frac{I + (B' \times T)}{T},$$

where B' represents the maximum number of bits that can be bypassed per second, and I is the system initialization constant. Another way of stating this assertion is that for time $T > 0$, $B \leq I + (B' \times T)$. Figure 4 illustrates this interpretation. Any continuous function

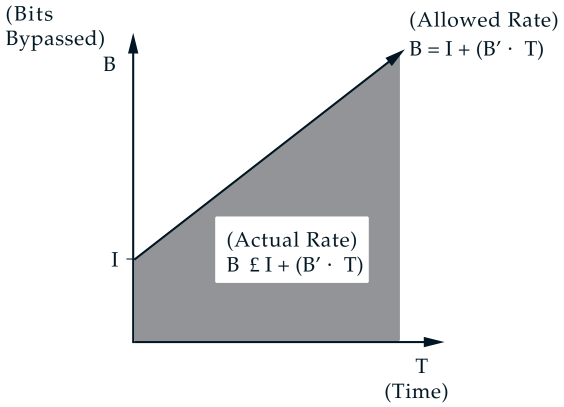

Figure 4: Constraining the bypass rate

in the shaded area could represent the actual bypass rate.

PRIORITY MESSAGE-ORDERING SCHEME. We decided to use a first in, first out scheme. This permits other ordering schemes to be enforced by the network if necessary while simplifying the design of the ECA.

Our Experience

Landwehr presented the following lessons learned from using the application-based modeling approach to formalize the MMS model [12].

- Stating even a carefully developed security model in a mathematical notation can be a substantial undertaking.
- It is important to keep not only the informal description of the model but also the application itself in mind when writing the formal description.
- Some properties that seem easy to assure in an implementation can be difficult to specify in a model.
- Be ready to adapt when hidden assumptions become apparent.

We considered this guidance when we specified the ECA model formally.

Additional lessons from our modeling effort are presented in this section. We first summarize the formalization of the ECA model and outline our approach for verifying that the ECA's implementation satisfies the formal model assertions. Henceforth, we use the term *assurance argument* to refer to the body of evidence that the implementation satisfies the model assertions.

ECA Model Formalization and Refinement

We chose the Communicating Sequential Processes (CSP) language [7] to formalize the ECA model. CSP permits the description of systems composed of networks of communicating processes. A CSP process communicates with its environment through named communication channels. Olderog and Hoare [20] describe a family of increasingly sophisticated models for CSP; less sophisticated members of the family enable specification and proof of a subset of properties that the more sophisticated members enable. We chose a particular model of CSP, called the Trace Model, because of its comparative simplicity and its ability to prove safety properties of networks of processes.

The Trace Model maps a process to an alphabet and a set of traces. The alphabet of a process specifies all communication events, i.e., channel-value pairs, in which it is permitted to engage. A trace of a process is an observation of its execution. It consists of a finite sequence of all communication events in which the process has engaged at some moment in time. Properties specified about systems described in CSP take the form of restrictions on the traces in which a process representing the system may engage. If the set of traces associated with the process actually conform to these restrictions, the process is said to satisfy the properties.

Figure 5 illustrates our process for constructing the assurance argument for the ECA's software.[4] Because of limited development resources, analysis of the hardware was beyond the scope of the formal assurance argument. The Software Cost Reduction (SCR) methodology [3, 22, 26]—which supports the exposition of the requirements, design, and implementation of software—provided a framework for expressing the assurance argument. The methodology's primary components are listed along the left side of Figure 5. Along the right side of the figure are the specification languages and tools (e.g., Statemate [6], mEVES and mVerdi [5], Verdix® Ada Development System (VADS®) [4]) that contributed to the implementation and verification of the ECA. The result of this collaboration between the SCR methodology and the specification languages and tools is the ECA's assurance argument, illustrated in the center of Figure 5.

The ECA assurance argument is not a simple refinement from the network security policy to the Ada implementation. A variety of formal and informal techniques were used to allow reasoning across four semantic domains, where each domain was chosen for its expressibility at a particular specification level. The network security policy was expressed in English,

[4] Details of the ECA software development process are described in [18].

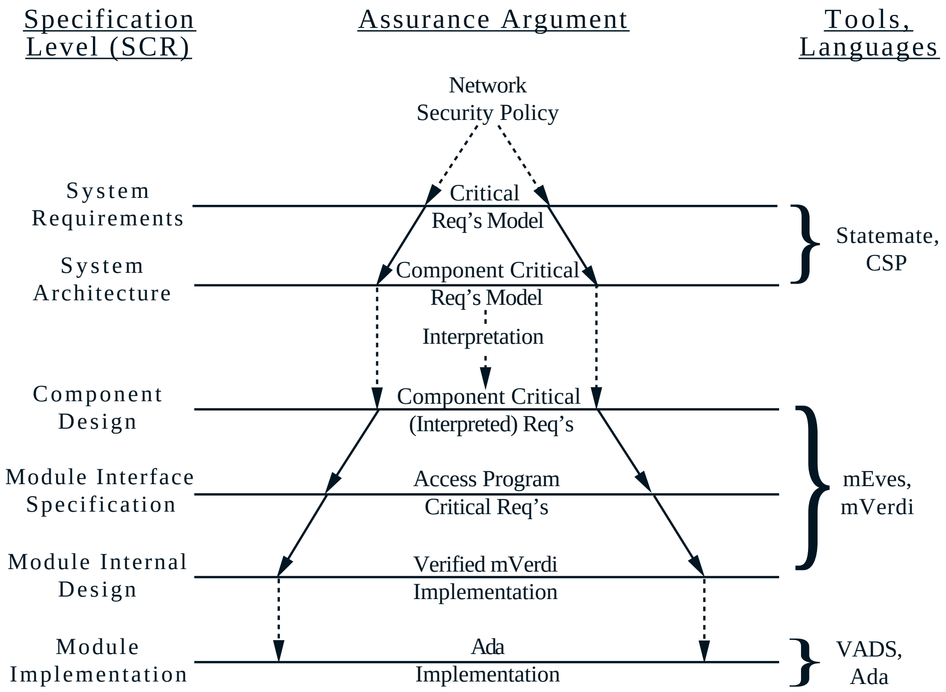

Figure 5: ECA Assurance Argument

the critical requirements model was specified and refined in CSP, the ECA's components were specified and verified in mVerdi and then implemented in Ada. Slanted arrows indicate a *refinement* of a specification to a more detailed specification or implementation; vertical arrows indicate a *translation* of a specification from one semantic domain to another semantic domain at a comparable specification level. Dashed arrows indicate a refinement/translation that is *informal*; solid arrows indicate a refinement/translation that is *formal*. The increase in width of the argument as we move from top to bottom illustrates additional detail that is specified at the lower levels.

The ECA was implemented as three communicating components: a *Red Processor* for processing sensitive data, a *Black Processor* for processing non-sensitive data, and the off-the-shelf cryptographic device. Twenty information-hiding modules make up the Red Processor and the Black Processor. The modules were proven formally to satisfy all but one of the ECA model's assertions (the *Correct Order* assertion was demonstrated using informal techniques). Complementary assurance techniques such as inspection, testing and simulation were also used.

Lessons Learned

1. Consider the application when choosing the modeling language.

One must consider the complexity of the application domain and requirements being modeled and the depth and rigor of the verification desired when choosing a modeling language. Our choice of the CSP Trace Model to specify the ECA requirements was based on a need to implement physical Red/Black (hardware) separation to facilitate certification, and a desire to allow (future) reasoning about the ECA and the network in which it is embedded.

While CSP served the assurance argument fairly well, it may not be appropriate for more complex applications or more complex critical requirements. The method of refinement in CSP is strictly procedural. There has been relatively little progress incorporating data or event refinement into CSP. This constraint forced us to use very similar data structures at the model and implementation levels and prevented us from rigorously verifying lower-level properties about the software and application-specific hardware that reside below the Trace Model abstraction, e.g., implementations of calls to send and receive messages over particular channels.

CSP also forced early decisions about the system's process structure, communication behavior, and method of data sharing (i.e., through message passing only). Other languages, e.g., [2], should be considered for applications in which the critical requirements are not naturally viewed as restrictions on communication behavior, the process structure and method of data sharing are not evident early on, or data refinement is necessary.

2. Postpone assigning parameter values that may differ among family members.

One motivation for defining the SBD domain and its associated modeling framework was the need to develop a series of increasingly sophisticated SBDs without knowing much about the operational characteristics of the environment in which they are to be embedded. To simplify ease of change/re-use of software specifications and code, the program family approach [21] encourages making design decisions that are shared by family members before making design decisions that differentiate family members. Since the family of SBDs to be built forms a subset of the SBD domain, this approach suggests delaying the instantiation of any parameter values that may differ between family members. As new family members are generated, we can reuse any part of the assurance argument that relies only on the shared decisions.

Unfortunately, this approach increases the complexity of the assurance argument development and certification for the first build. As a result, the approach was impractical for the ECA development. The ECA's tight development schedule forced us to assign the model parameters according to the specifics of the ECA's environment (e.g., fixed key, FIFO priority) and to simplify the corresponding assertions. If future maintenance of the ECA (or a subsequent build) demands a capability inconsistent with a parameter value, a complete re-implementation and re-analysis may be necessary. In fact, this limitation was realized in a subsequent build of the ECA by a developer considering over-the-network re-key. Our effort suggests that when planning for the development of a family of systems, managers and developers need to allocate sufficient time up-front for constructing an assurance argument that can be easily modified to apply to different family members.

3. Expect unforeseen security-relevant flows during refinement.

The confidentiality assertions of the ECA (and SBD) Model are defined in terms of the primitives of the external interface that are visible at the model level of abstraction. Although the CSP Trace Model guarantees that any proper refinement of these assertions to a lower level specification or implementation also satisfies these assertions, it does not guarantee that any information flows introduced as a result of the refinement satisfy the intent of the assertions, i.e., to enforce **Restricted Red-To-Black Flow**. Thus, the refinement may be less secure with respect to **Restricted Red-To-Black Flow** than the system description at the model level of abstraction. Other work [13, 16, 17] describes in more detail the problems associated with preserving information flow security properties using conventional system refinement techniques.

We understood the limitations of conventional system refinement techniques at the start of the ECA development project, but we believed that we could specify the ECA external interface abstractly and without omitting information that was critical to analyzing the security of the system. Our approach was to model all information flow with external processes, including both message traffic and system control, as communications over CSP channels. The critical requirements were then formulated in terms of these information flows. We believed that if all information flows were in fact modeled, any refinement of the model should be as secure as the model with respect to **Restricted Red-To-Black Flow**. The external interface was implemented in a hardware hiding module as calls to *put* and *get* messages over channels. It became clear, as the details of the external interface were negotiated with the end users of the ECA, that information flows beyond those that were modeled were needed to implement the communication protocol between the ECA and its external environment.

In hindsight, it was unreasonable to assume that there would not be additional security-relevant flows introduced in the implementation of the hardware hiding module. Recent work on the nature of possibilistic security properties [17] discredits our initial belief that if all security-relevant information flows are explicit in the model then any refinement is as secure as the model. The ideal solution to the problem is to use security-preserving refinement techniques, e.g., [8, 15], combined with an information flow security model. Unfortunately, these techniques are not yet practical for real system development. Until practical security-preserving refinement techniques become available, conventional refinement techniques, such as those for CSP, can be used to develop secure systems. These techniques must be augmented with an analysis of information flows introduced during refinement, similar to to the covert channel analysis required by the TCSEC [19]. This analysis has yet to be performed for the ECA.

4. The choice of abstractions is critical to the coherence of the assurance argument.

One of the biggest lessons that we learned about the overall ECA development process is that, with respect to independent system certification, coherence of presentation is at least as important as rigor in specification and verification. We have started investigating the use of literate programming techniques [9] as a means to present and manage specifications and verifications in a more coherent and intuitive manner, but this, by itself, is not sufficient. The abstractions chosen to represent system primitives in the top-level specifications need to be clear enough for a third party to understand (and agree on their use), yet precise enough to permit formal verification.

Determining whether the abstractions are appropriate is a difficult process. The developer and the users need to agree early on the primitives of the user interface and on the critical requirements, but the model should not be fixed until all relevant issues are resolved. For example, the ECA model was baselined prematurely and then significant progress was made implementing and formally verifying the module that implements the format check associated with *Correct Format*. Unexpected changes to the primitives of the ECA's external interface decreased the efficiency of the module and the understandability of its formal verification. An independent inspection of the module lead to its rejection even though it had been formally verified! The current formulation supports a more precise and intuitive refinement.

Acknowledgments

The authors wish to thank Ken Hayman, DSTO Australia, for his contribution to the ECA development effort and Ran Atkinson, Carl Landwehr and John McLean of NRL, Clark Weissman of Paramax and the anonymous reviewers for their helpful comments and suggestions.

References

[1] D.E. Bell and L.J. La Padula. Secure computer system: Unified exposition and Multics interpretation. Mitre Technical Report MTR-2997, Mitre Corp., Bedford, MA, March 1976.

[2] K. M. Chandy and J. Misra. *Parallel Program Design: A Foundation*. Addison-Wesley, 1988.

[3] Paul C. Clements. Using information-hiding as a design discipline: Techniques and lessons. In *Proc. Structured Development Forum VIII*, Seattle, WA, August 1986.

[4] Verdix Corporation. *Verdix Ada Development System (VADS) Version 6.0 Manual Set*, 1990.

[5] Dan Craigen. m-EVES. In *Proc. 10th NBS/NCSC Computer Security Conference*, pages 109–117, Fort George Meade, MD, September 1987. National Computer Security Center.

[6] David Harel, Hagi Lachover, Amnon Naamad, Amir Pnueli, Michal Politi, Rivi Sherman, Aharon Shtull-Trauring, and Mark Trakhtenbrot. Statemate: A working environment for the development of complex reactive systems. *IEEE Transactions on Software Engineering*, 16(4):403–414, April 1990.

[7] C. A. R. Hoare and J. C. Shepherdson, editors. *Mathematical Logic and Programming Languages*. Prentice-Hall, 1985.

[8] Jeremy Jacob. On the derivation of secure components. In *Proc. IEEE Symposium on Security and Privacy*. IEEE, 1989.

[9] Donald E. Knuth. Literate programming. *The Computer Journal*, 27(2), May 1984.

[10] C. Landwehr, C. Heitmeyer, and J. McLean. A security model for military message systems. *ACM Transactions on Computer Systems*, 2(3):198–222, August 1984.

[11] Carl E. Landwehr. Assertions for verification of multilevel SMMS. *ACM SIGSOFT Software Eng. Notes*, 5(3):46–47, July 1980.

[12] Carl E. Landwehr. Some lessons from formalizing a security model. In *Proc. Verkshop III* in *ACM Software Engineering Notes*. Association for Computing Machinery, August 1985.

[13] Daryl McCullough. Noninterference and the composability of security properties. In *Proceedings of the 1988 IEEE Computer Society Symposium on Computer Security and Privacy*, pages 177 – 186, Oakland, CA, 1988.

[14] John McLean. The algebra of security. In *Proc. 1988 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, April 1988.

[15] John McLean. Proving noninterference and functional correctness using traces. *Journal of Computer Security*, 1(1), 1992.

[16] John McLean. Models of confidentiality: Past, present and future. In *Proc. Computer Security Foundations Workshop VI*, Franconia, NH, 1993. IEEE Press.

[17] John McLean. A general theory of composition for trace sets closed under selective interleaving functions. In *Proc. 1994 IEEE Symposium on Research in Security and Privacy*, May 1994.

[18] Andrew Moore, Eather Chapman, David Kim, Eric Klinker, David Mihelcic, Charles Payne, Maria Voreh, and Ken Hayman. External COMSEC adaptor software engineering methodology. NRL Memorandum Report (to appear), Naval Research Laboratory.

[19] National Computer Security Center, Ft. Meade, MD. *DoD 5200.28-STD, Trusted Computer System Evaluation Criteria*, December 1985.

[20] E.R. Olderog and C.A.R. Hoare. Specification oriented semantics for communicating sequential processes. *Acta Inform.*, 23:9–66, 1986.

[21] D. L. Parnas. On the design and development of program families. *IEEE Transactions on Software Engineering*, SE-2(3):1–9, March 1976.

[22] David L. Parnas, Paul C. Clements, and David M. Weiss. The modular structure of complex systems. In *Proc. 7th Internation Conference on Software Engineering*, pages 408–416. IEEE, March 1984.

[23] Charles N. Payne, Judith N. Froscher, and Carl E. Landwehr. Toward a comprehensive INFOSEC certification methodology. In *Proceedings of the 16th National Computer Security Conference*, pages 165–172, Baltimore, MD, September 1993. NIST/NSA.

[24] Charles N. Payne, David M. Mihelcic, Andrew P. Moore, and Kenneth J. Hayman. The ECA critical requirements model. NRL Formal Report 9528, Naval Research Laboratory, Washington, D.C., December 1992.

[25] David Sutherland. A model of information. In *Proceeding of the 9th National Computer Security Conference*, September 1986.

[26] A. John van Schouwen, David Lorge Parnas, and Jan Madey. Documentation of requirements for computer systems. In *Proc. IEEE International Symposium on Requirements Engineering*, San Diego, CA, January 1993. IEEE.

[27] Victor L. Voydock and Stephen T. Kent. Security mechanisms in high-level network protocols. *Computing Surveys*, 15(2):135–171, June 1983. Also in Rein Turn, *Advances in Computer Security*, Volume 2, Artech House, 1984.