

## **Conflicts Between Security/Survivability Requirements and Other Types of Requirements: How Do They Arise and Can They Be Resolved?**

### **Panel Chair:**

Nancy R. Mead, SEI/CERT

### **Panelists:**

Constance Heitmeyer, Naval Research Laboratories

Richard C. Linger, SEI/CERT

Gary McGraw, Cigital

### **Panel Description:**

Nancy R. Mead  
Software Engineering Institute  
4500 Fifth Ave., Pittsburgh, PA 15213  
E-mail: nrm@sei.cmu.edu

Requirements for security and survivability, when they exist, are often specified independently of other requirements. As a result the various types of requirements frequently conflict with one another.

- Security requirements may be developed as an add-on after other functional and non-functional requirements are already in place. In this case, the result may be a set of requirements that reflect a fortress mentality. The conflicts inherent in this situation are often 'solved' by deciding to add more firewalls.
- Many projects call for maximizing the use of COTS, but the security strategies and implications of COTS are seldom discussed.
- Systems may have performance requirements and security/survivability requirements that inherently conflict with one another, but if these requirements are not analyzed side-by-side, such conflicts may not be recognized until late in the development process, when resolution is difficult.

Methods and processes are needed that will allow security and survivability requirements to be developed in conjunction with other requirements, and that will allow the needed tradeoffs to take place. Many of today's methods traditionally focus on functional requirements and de-emphasize other types of requirements.

This panel will discuss and debate

- The relationship and conflict between security/survivability requirements and functional requirements
- The relationship and conflict between security/survivability requirements and other non-functional requirements
- How to set priorities between various requirement types
- How to develop security and survivability requirements so that they are not an add-on patch to a set of existing requirements
- Methods and processes that will support development of security and survivability requirements

## Panelist Position Statement 1:

### Defining Security and Survivability Requirements\*

Constance Heitmeyer  
Naval Research Laboratory (Code 5546)  
Washington, DC 20375  
E-mail: heitmeyer@itd.nrl.navy.mil

In *high assurance systems*, compelling evidence is required that the system delivers its services in a manner that satisfies certain critical properties. Among these properties are *security properties*, which prevent unauthorized disclosure and modification of sensitive information, and *survivability properties*, which allow the system to continue its mission despite malicious attacks, accidents, or failures. While a secure system generally prevents bad things from happening (e.g., prevents an unauthorized user from reading a classified document), a survivable system causes good things to happen (e.g., provides critical user services despite a malicious intrusion or a hardware failure). An important issue in system development is how to define the requirements of systems that must be both secure and survivable.

Beginning in the 1970s, the Bell-LaPadula model [1] was commonly used to define the security requirements of systems. This model describes system security in terms of two properties: the *simple security rule* (a subject cannot read an object for which he is not cleared) and the *\*-property* (a subject cannot move information from an object with one security classification to an object with a lower classification). In 1984, an alternative application-based approach to defining security requirements was proposed; the MMS (Military Message System) security model illustrated this approach [4]. Rather than defining security generally and focusing at the operating system level, the MMS model defined security in terms of the application, describing constraints on user access to messages and message files and constraints on other user operations, such as message downgrading and message release. In subsequent years, the security requirements of many other systems (see, e.g., [5,2]) have followed the example demonstrated in [4], defining security in terms of application requirements rather than operating system structure.

In my view, survivability, like security, is best described in terms of an application. One approach to defining the requirements of a survivable system is to assign each system service with a survivability measure. The services tagged with the highest measure are the minimal set of services the system needs to continue its mission. How to assign survivability measures to system services and how to transition from a system supporting the full range of services to a system providing a service level that is degraded but still sufficient to support the current mission are research issues. Another important research question is how to use the survivability measures and knowledge about system resources to dynamically select the services to be supported when malicious attacks or failures occur.

At least two approaches to implementing a survivable system can be identified. In the first approach, the system detects and dynamically replaces system components damaged by an attack or a failure. Such an approach could be implemented by a distributed agent-based system. An alternative approach makes services tagged with high survivability measures less vulnerable to malicious attacks than services with lower measures. Among the services likely to be assigned high survivability measures are *security services*, e.g., services that support confidentiality, integrity, etc. Typically, such services are implemented using a single method; for example, in a system that offers secure communications, confidentiality may be provided by DES and integrity

by keyed MD5. However, a security service supported by a single method is highly vulnerable to a malicious attack. To enhance the survivability of security services, the designers of the Cactus framework use redundancy [3]. For example, in Cactus, confidentiality is achieved by using a combination of cryptographic algorithms and a variety of methods for defining keys.

## References

- [1] D.E. Bell and L.J. LaPadula. Secure computer system: Unified exposition and Multics interpretation. MTR-2997, MITRE Corp., Bedford, MA, 1976.
- [2] C. Heitmeyer, M. Archer, J. Kirby, and E. Leonard. Formal security policy model and formal specification for the PEIP II cryptographic device. Draft NRL report, Wash., DC, 2002.
- [3] M. Hiltunen, R. Schlichting, and C. Ugarte. Enhancing survivability of security services using redundancy. In *Proc., Intern. Conf. on Dependable Systems and Networks (DSN 2001)*, Gothenberg, Sweden, 2001.
- [4] C.E. Landwehr, C.L. Heitmeyer, and J. McLean. A security model for military message systems. *ACM Trans. on Computer Systems* 2(3): 198-222, August 1984.
- [5] C. Payne, J. Froscher, and C. Landwehr. Toward a comprehensive INFOSEC certification methodology. In *Proc., 16th National Computer Security Conf.*, Baltimore, MD, 1993.

\* This work is supported by the Office of Naval Research.

## Panelist Position Statement 2:

Richard C. Linger  
Software Engineering Institute  
4500 Fifth Ave., Pittsburgh, PA 15213  
E-mail: rlinger@sei.cmu.edu

Government, defense, and business enterprises are totally dependent on large-scale network systems whose complexity frequently exceeds current requirements engineering capabilities. The result has been persistent difficulties in system development, management, and evolution, and failures, intrusions, and compromises in operation. These systems are composed of very-large-scale, heterogeneous networks with often-unknown boundaries and components. Dynamic interconnectivity of federated systems limits visibility and control of security and survivability. User task flows can traverse multiple systems and domains with varying security and survivability characteristics. And these systems must deal with uncertain COTS component function and quality, unforeseen behaviors and vulnerabilities, and unanticipated cascade failures. Complexity is compounded by the asynchronous and virtually unknowable interleaving of communications among system components. Requirements engineering in this environment is a challenging proposition.

How systems behave when they fail has become as important, and possibly more important, than how they behave when they succeed. Large-scale network systems are permanently and unpredictably subject to failures, intrusions, and compromises. User task flows can engage in extensive traversals of network nodes and communication links, where the behavior of invoked services cannot always be known or predicted. In this environment a variety of uncertainty (survivability) factors must be addressed in requirements specification, including:

- unpredictable function: A service may be provided by COTS components of unpredictable or unknown function and reliability, and may not perform expected operations.
- compromised function: A service may have been compromised or disrupted by an intrusion or physical attack and may not be able to perform its function correctly or at all.
- modified function: A service may be modified or replaced as part of routine maintenance, error correction, or upgrade, with intentional or inadvertent modification of its function.
- asynchronous function. A service may be employed simultaneously by other users, and thus produce results dependent on unpredictable history of use, both legitimate and illegitimate.

Thus, modern requirements engineering must create behavior specifications for both normal-mode operations and failure-mode uncertainties on an equal basis. These specifications must define required behavior in all circumstances of use, both desired and undesired, and integrate functional and non-functional (quality attribute) requirements in a seamless manner at all levels

of abstraction.

Mathematical foundations for requirements engineering must define semantics that accommodate uncertainty factors as an intrinsic characteristic of systems. Rigorous semantic models are necessary that prescribe requirements engineering practices for addressing the uncertainty factors. In short, integrated semantic models for requirements engineering will permit unified treatment of normal and failure operations and their associated quality attributes. Fundamental research is required to develop the underlying semantics for these models.