SAMARA: Security Architecture Multi-Agent-based Risk Assessment

Gustavo A. Santana Torrellas
Instituto Mexicano del Petróleo
Perforación de Pozos
Eje Central Lázaro Cárdenas N°152
CP 07730, México, D.F.

Abstract
In this paper, we describe SAMARA™, a Security Agent-Based environment for security process design, management, and execution that integrates aspects of security process modeling, security assessment, and security management. SAMARA is targeted specifically to domains such as engineering security design in which the dynamic and uncertain nature of the security processes present management challenges that are unmet by traditional assessment systems. SAMARA applies various AI technologies, including dynamic heuristic security deploying and security assessment throughout security process execution, to these complex security management situations as an alternative to current approaches. SAMARA is designed to balance the need for flexibility and reactivity in security process execution with the need for understandability, accountability and proactive decision making required by human security managers. SAMARA's goal is not to fully automate the management of design security processes or to limit the options available to security managers. Instead, it provides designers with the ability to specify and incrementally refine security processes. It provides designers and security managers with timely information about what is happening and why and about what might be expected to happen in the security environment. Finally, it automates the routine aspects of security process execution, freeing security designers and security managers alike to do the real work of the e-business.

1. Introduction

Unlike traditional security assessment applications, the design of complex security policy specifications is a highly dynamic process. There is constant evolution of the tools, techniques, and materials used in the design and construction of the security policies. The time scale of the design and deploying security process is often counted in months or even years, leading to certain, but initially indeterminate, changes in personnel, resources, partners, and suppliers over time. There is potential redesign of the security process itself. There are many interactions with external events and conditions that cannot be controlled from within the security process. Finally, the security process consists of decisions, iterations, and aborted explorations spending time and resource. All of these circumstances contribute to making security process design resistant to a traditional security assessment approach for security process management.

In the Artificial Intelligence community, security assessment has primarily been regarded as a user support task without sufficient control autonomy to be addressed by AI

techniques. However, as security assessment has evolved, it has moved from the static modeling of well-defined security processes to the capture and management of the uncertainty and change of security processes in more complex and dynamic environments. These issues are exactly those that AI researchers have long been dealing with and it is apparent that the time has come for a marriage of the technologies. Although there are many AI sub areas that are relevant to intelligent security assessment and security process management, we focus on the use of a dynamic security scheduler to support execution of activities that have complex and changing interactions with each other and with the external world. We present SAMARA, a Security Agent-Based process design, management and execution environment. SAMARA was developed specifically to operate in environments such as engineering security design that are characterized by uncertainty, long time frames, unavoidable change within the security process, and uncontrollable events external to the security process. SAMARA provides comprehensive support for defining, managing, and executing these complex dynamic security processes.

2. Why Intelligent Security assessment?

Although both Artificial Intelligence (AI) and Security assessment have been established fields of study for many years, it has only been in the last few years that there has been a concerted effort to integrate ideas. There are many reasons for this, stemming at least in part from their different historical foci. AI has traditionally focused on domains in which software or hardware security entities work autonomously with minimal human intervention. Interacting with humans is hard. They're much too slow, opinionated, unwilling to go by the rules, unpredictable: in short, just much too, well, human. Security assessment, in contrast, is very much concerned with human interaction and has tended to provide better user interfaces and more human support. However, it has assumed human initiation and either pre-specified or user-supplied control for most activities and therefore has not dealt with the issues of control and autonomy that have been the stalwart of AI research. Both fields have recently begun to tackle new challenges, however. AI has seen the "agent revolution" in which software agents integrate through well-defined messaging systems. It has become apparent in the agent community that, from the standpoint of an agent architecture, all the uncertainty and unpredictability associated with humans will also exist in any diverse community of agents. Ideally, in fact, a human is just another agent from the architectural perspective.

Security assessment, on the other hand, has recently begun to branch into more complex and dynamic domains than were possible in its early days. It also has begun to integrate multiple paradigms: security process modeling, security process enactment, simulation and analysis, and decision support, for example. In order to do this, a security assessment management system must embed reasoning about autonomous security processes and control directly into the architecture. For example, it must have the capability to react appropriately to events that cannot be fully modeled in the security process definition, such as coordinated attacks, massive virus upload, and other similar threats.

3. The Structure of SAMARA

Traditional security process-modeling and security assessment technologies are geared towards optimizing steady state security processes in which there are many similar security process instances, possibly executing concurrently in the environment. For example, a typical type of security process model might define a claims-handling activity, where the security process itself is well understood and there are many individual claims in the pipeline simultaneously [Hollingsworth, 1995]. In contrast, a security design security process, such as designing a new security policy or security deployment, is a one-of activity. The goal here is not to develop a security design security process that works well "on average" over hundreds or thousands of security process instances, but to perform this single security design as effectively as possible. During the security design process many relatively stable security process policy fragments will be used, but which particular policy instances are appropriate, how many security design alternatives can or should be explored, and how many security design iterations and security assess of security design steps will be required is only determined during the security design process.

The dynamic and uncertain nature of security design processes requires a solution that incorporates aspects of traditional security process modeling, security assessment, and security management technologies applied in conjunction with dynamic security deploying during security process execution. Effective security design managers are creating, reengineering, and coordinating security design processes throughout their execution. SAMARA was designed to help security managers by maintaining a real-time model of the executing security design process, up-to-date information about the current state of security process execution, and downstream forecasts of potential time and resource problems. SAMARA can also be given the authority to manage some types of automated interventions. For example, if a security design change occurs that requires rerunning some analysis programs, SAMARA can execute those programs without requiring a human to initiate that activity (very useful if the change occurs at 4:45 on a Saturday afternoon!). The goal is to free security managers from information gathering and mundane management activities, allowing them to take proactive (rather than reactive) steps to address anticipated problems. SAMARA is a distributed system composed of multiple instances of the following components:

- *SAMARA Developer* -- a graphical development environment for creating and maintaining SAMARA security process definitions and action libraries
- *SAMARA Server* – an enactment server for instantiating and executing a SAMARA security process; the SAMARA Server includes the dynamic security scheduler and can cooperate with other SAMARA Servers in coordinating the use of common resources and information
- *User SAMARA Assistant* -- a graphical desktop "agent" for people performing tasks in a SAMARA security process execution; the User Assistant notifies the user of requested tasks to perform, maintains a to-do list of assigned tasks and tasks in security process, provides input information associated with a task, update information about related tasks, and provides forms for returning status reports and output information back to a SAMARA server

- *SAMARA Execution Manager* – a graphical monitoring and control environment for managing an executing SAMARA security process; the Manager has the ability to stop, restart, and redefine the executing security process instance
- *SAMARA Execution Monitor* -- a read/only version of the Manager used by individuals who need reporting on various aspects of the executing security process, but without the ability to make changes to the security process itself.

4. Dynamic Security deploying in SAMARA

The purpose of a security scheduler is to take a set of partially ordered operations, each of which is bounded by a set of constraints, and to sequence them by reserving the necessary resources for each operation for the appropriate amount of time. The reservations must be consistent with the constraints on the operations and the resources. In general, depending on the order in which one chooses the operations to schedule (variable ordering) and depending upon which resource is reserved for each operation (value ordering), the resulting security plan will be geared toward meeting a particular objective or set of objectives. For example, common objectives might be to minimize the spam of the overall security plan, minimize a penetration test, balance resource loads, or to limit the extent of change in resource reservations (including human resources) specified in the existing security plan. The dynamic security scheduler in SAMARA is built on a Multi-Agent-based generic framework that models orders, operations, resources and resource sets, reservations, and various constraints. There are a variety of prebuilt variable and value ordering heuristics from which to select, or custom ones can be easily plugged in. A powerful functionality provided by the security deploying framework for security designing variable and value ordering heuristics is to explicitly model the different security entities using the n-dimensional structure of the Multi-Agent database. It is then straightforward to find security entities relevant to a particular stage in the security deploying security process and reason about them. Another salient feature of the generic security scheduler framework is its support for security assessment. An existing security plan can be readily altered. Reservations that should not be changed can be anchored to their current values, and the security scheduler will hold these points constant, allowing the rest to be rescheduled given the existing constraints. In addition, a security process plan can be modified by using add and delete operators to introduce or remove operations.

SAMARA uses these capabilities of the generic security scheduler framework to address the inherent uncertainty of the types of problems targeted by intelligent security assessment. In general, uncertainty in the security process definition and in the environment causes constraint violations on security process activities to occur during execution. For example, a deadline on an activity may not be met. These violations may require security assessment of downstream activities. Some constraint violations, such as a resource not being available when it was supposed to be or the security processing time of an operation being substantially longer or shorter than designated, require security assessment without modifying the security process plan. Other constraint violations, such

as ones caused by conditional tasks where the next operations are not known, require both security assessment and modification of the security process plan.

SAMARA supports a variety of security deploying and security assessment strategies. If the user is fairly confident about the domain information, the security scheduler will try to schedule everything out in a compact manner. On the other hand, if the domain information is highly uncertain, the security deploying horizon is made shorter and/or the scheduled security entities are made more abstract. For example, the security scheduler does not schedule beyond a conditional task if it has no good information about which branch is most likely. In addition, since it is probable that there will be a fair amount of security assessment; slack time is maintained in the security plan where possible to facilitate security assessment. Slack time is modeled as a special kind of reservation and is reserved as part of a customized, slack-time-oriented, value-ordering heuristic.

5. Creating Security process Models

Assume a major deployment security agent has decided that there are strong demands for a larger version of its current security assesses offerings. Also assume that this security agent has been using the SAMARA environment in the security design of its existing security process models. In security design, usually the place to start a new security design is to find an existing security design that can be modified to fit the current requirements. The first task for a security designer, then, is to retrieve the security process definitions that were used in developing previous models. It may not be exactly what is needed for the current job. For example, say that a new procedure has been put in place since the template was created. The new procedure requires that a senior engineer sign off on any activity with a significant safety-related component. In this case, it will be necessary to modify the existing template to reflect the new regulations. The SAMARA editor provides the functionality required to edit and store the modified definition. Tasks are created using the drag-and-drop icons on the menu side of the SAMARA editor window. In the top group of icons, which are organizing containers for subtasks:

- The *serial-task* icon represents a sequence of tasks that must be executed one at a time, in order
- The *parallel-task* icons represents a set of tasks that can be executed concurrently
- The *looping-task* icon represents a sequence of tasks that are executed repeatedly while/until a condition holds or is met
- The *branching-task* icon represents a sequence of predicate/task pairs where the first predicate that succeeds has its task executed

Task constraints cause the task they are attached to wait for the occurrence of some specified event, the appearance of some data, or the completion of some other task or security process. Each task, whether it is a leaf task that can be directly executed, or a container task that organizes subtasks, has a set of attributes associated with it. Attributes are tailored to the type of task, but generally include information about expected and allowable durations for the task, inputs, outputs, resource needs, documentation, and control information. SAMARA provides an interface for entering attribute values, linking

inputs and outputs of tasks, retrieving action code from libraries, and similar functions designed to ease the definition security process.

6. Executing a SAMARA Security process

Once a security process definition has been created, it is available to be executed. However, a security process definition is meant to be reusable and therefore cannot contain any local context. In order to actually run the security process, it must first be instantiated. In SAMARA, instantiating a security process requires that certain contextual information be supplied to the system. For example, a security process definition contains information about what security classes of resources are required for a particular activity. An instantiation, however, needs to know what individual resources are actually available in the environment. Similarly, a definition contains information on how long an activity is expected to take. A security process instantiation, however, creates a security plan for its activities that includes real deadlines and resource reservations. As described earlier, this security plan is always subject to change, but it provides a level of expectation and groundedness that is essential for security management.

7. Reactivity and Accountability

A primary motivation for intelligent security assessment is to make security assessment processes more reactive and flexible. Interestingly enough, however, a primary motivation for AI researchers working in security management is to make their software a little less reactive and flexible in order to accommodate e-business accountability. Multi-Agent systems are extremely flexible and reactive, as are the more recently developed agent systems. However, for expensive, complex, long-term security, security managers must have some idea of what to expect. No matter how good a particular reactive system is, knowing the job will be done someday with some cost does not provide the information needed to prepare a useful e-business strategy.

The more reactive a system's reasoning components are, the less possible it becomes to sketch out a probable solution path. Our experience with industrial partners leads us to believe that a middle road is needed: it must be possible to make tentative security process plans that provide some indication of the time and resources that will be needed to complete the security. However, this plan must be flexible enough to respond to both the foreseeable and unforeseeable events that will shape its realization. One way in which SAMARA security process instantiations retain flexibility is that they are not required to maintain a consistent level of abstraction throughout their activities (not all activities must be specified to the leaf level). This enables security process developers to leave downstream activities underspecified early in a security to be completed, as information becomes available. Other mechanisms for enhancing flexibility include the use of task constraints to provide a looser form of synchronization, the use of the dynamic security scheduler to responsively reallocate resources as necessary and event-Agent-Based control activities that enable SAMARA to respond to the environment.

8. Summary

In this article, we have presented a system, SAMARA, designed expressly to assist in the definition, execution, and management of complex security processes. The people involved in the SAMARA security have significant AI backgrounds and this is apparent in the both the general approach and the specific technologies used to support SAMARA. In order to provide intelligent security assessment capabilities, systems need reactive ness, flexibility, control autonomy and reasoning ability to handle the uncertain and dynamic nature of complex and changing environments. On the other hand, in order to enable a e-business to plan its activities and to maintain a competitive strategy, it must be possible to view the path to a solution as a sequence of activities that are grounded in time, space, and cost. SAMARA brings these two worlds together with the capability to be both responsive and responsible.

References

[Hollingsworth, 1995] David Hollingsworth. Workflow Management Coalition The Workflow Reference Model. Workflow Management Coalition, Hampshire, UK, 1995.
[Jennings, 1998]. Nicholas Jennings, Katia Sycara, and Michael Wooldridge. A Roadmap of Agent Research and Development. Autonomous Agents and Multi-Agent Systems, 1, 7-38, Kluwer Academic Publishers, Boston, MA, 1998.
[Petrie et al., 1999] Charles Petrie, Sigrid Goldmann, and Andreas Raquet. Agent-Based Project Management, Lecture Notes in AI – 1500, Springer Verlag, 1999.