SARA: Survivable Autonomic Response Architecture

Scott M. Lewandowski, Daniel J. Van Hook, Gerald C. O'Leary,
Joshua W. Haines, and Lee M. Rossey
Lincoln Laboratory, Massachusetts Institute of Technology
244 Wood Street
Lexington, Massachusetts 02420-9108
{scl,dvanhook,oleary,jhaines,lee}@sst.ll.mit.edu

Abstract

This paper describes the architecture of a system being developed to defend information systems using coordinated autonomic responses. The system will also be used to test the hypothesis that an effective defense against fast, distributed information attacks requires rapid, coordinated, network-wide responses. The core components of the architecture are a run-time infrastructure (RTI), a communication language, a system model, and defensive components. The RTI incorporates a number of innovative design concepts and provides fast, reliable, exploitation-resistant communication and coordination services to the components defending the network, even when challenged by a distributed attack. The architecture can be tailored to provide scalable information assurance defenses for large, geographically distributed, heterogeneous networks with multiple domains, each of which uses different technologies and requires different policies. The architecture can form the basis of a field-deployable system. An initial version is being developed for evaluation in a testbed that will be used to test the autonomic coordination and response hypothesis.

1. Introduction

Information assurance tools are designed to defend information systems against malicious adversaries. While information assurance has been studied for many years, a new class of threats involving high speed and broad scale attacks has only recently become the subject of formal research. To provide consistently high service quality and to ensure mission success, an automated system is required to counter these threats since humans are not fast enough to react to high speed or broad scale attacks effectively.

The DARPA/ISO Autonomic Information Assurance (AIA) program studied these new attacks and possible ways to defend against them. Two hypotheses resulting from the AIA program are: that fast responses are necessary to counter advanced cyber-adversaries; and that coordinated responses are more effective than local reactive responses.

This paper describes an architecture for autonomic information assurance systems and discusses the requirements for the decision-making and communication that is required to support autonomic information assurance. Since no existing system meets all of the communication needs of autonomic information assurance systems, the Survivable Autonomic Response Architecture (SARA), an architecture that does meet these needs, is presented. SARA is able to support the decision-making functions discussed, although it makes no direct contribution to this area. The SARA architecture was designed for information assurance applications, but it is also well-suited for use in a wide variety of communication systems outside of the information assurance domain. Although the architecture is designed for field-deployment, its immediate role will be to support experimentation that confirms or refutes the two hypotheses listed above.

1.1. Autonomic response

Responses taken automatically by a system without real-time human intervention are autonomic responses. The term "autonomic" is used by analogy to the autonomic nervous system, which automatically controls and regulates many motor and physiologic functions without conscious input from the host. Instead of relying on human input while carrying out the response, autonomic information assurance systems rely on human knowledge and policy that is programmed into the system in advance. A perfect autonomic response system would

This work was sponsored by DARPA under Air Force Contract F19628-00-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Air Force.

ISBN 0-7695-1212-7/01 $10.00 (C) 2001 IEEE

be able to make rational decisions consistent with goals set by humans. The autonomic system should be capable of making decisions that help the system meet the goals much more quickly and accurately than a human could.

A system call wrapper is a simple example of an autonomic response system. The wrapper intercepts system calls made by an application program, thus mediating between the application program and the operating system. When an application makes a system call, the wrapper assumes control and is able to inspect the parameters passed to the function and the current operating context. This information can be used to match the observed call to one or more pre-programmed policies. Policies implicitly specify what action the wrapper should take. For example, the wrapper could allow the system call to proceed per usual, it could modify one or more parameters before executing the system call, it could fail the system call by returning an error code to the calling application, or it could kill the calling application. A simple policy might be to forbid applications from accessing files in certain directories. When an application tries to access the forbidden directories, the wrapper could fail the system call by returning an "access denied" error to the application.

Simple pre-programmed responses to attacks may mitigate the direct impact of well-known attacks, but they have some undesirable properties. These "knee-jerk" responses are predictable and potentially exploitable by an adversary. An adversary may be able to circumvent an expected response. It is also possible that an adversary could induce the system to compromise service quality to legitimate users by triggering the response through some carefully crafted, but otherwise benign, action that the response mechanism interprets as malicious.

Simple pre-programmed responses are weak because they are selected based on local information, which provides only a small number of specialized inputs; global state is not used by the selection heuristic. Although a chosen response may be locally optimal, the dearth of information limits the ability of a local response selection system to choose the response that most benefits the entire system. Many biological autonomic response systems also suffer from the inability to account for global system awareness. For example, the human nervous system responds to many threats based only on local information, without regard to global situational awareness. An example is the reflexive response of pulling away from a hot object. This response happens automatically without taking into account, for example, other objects that could be struck while pulling away. It is also important to incorporate global knowledge when choosing local reactions since those local reactions can impact other system components. For example, a local response selector may decide to block traffic entering and leaving its host. If the host is functioning as the DNS server for a network, the global impact of the local response would cripple the ability of the hosts on the network to locate the resources they need to perform the mission.

1.2. Coordinated autonomic response

Although responses selected based on local information are useful in many cases, autonomic responses to attacks may be even more effective if they were selected based on global knowledge that is not available at the local response site. Global information could be used to select the responses that best contribute to an overall strategy for maintaining service quality. Selecting optimal responses requires coordination among the components gathering data about and defending an information system. Specifically, a decision-making entity must be able to gather information about the state of a mission and its supporting system to determine which countermeasures should be used to counter malicious activity. When choosing countermeasures to be deployed, a response selection system must account for the manner in which resources will be affected and the resulting impact on the mission. To effectively select a response, a decision-making entity must weigh the costs and benefits of all of the possible responses. In addition, special attention must be given to the possibility that the event being responded to is a misdetection; the ensuing repercussions of acting based on inaccurate data must be carefully considered.

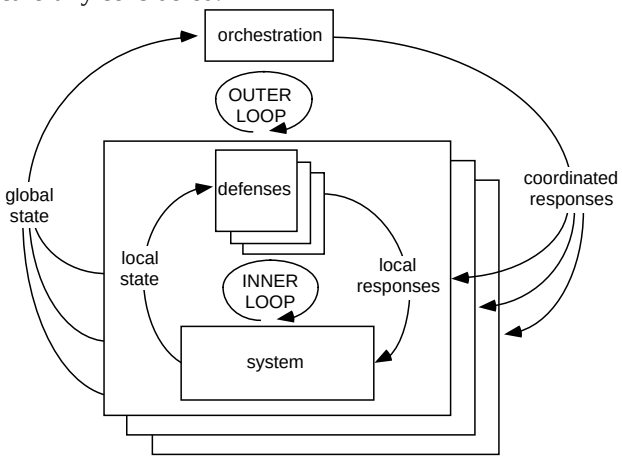

Figure 1 Inner and outer loops.

Response systems that rely only on local information are able to act very quickly since they do not need to wait for information to be gathered and relayed before making a decision. In addition to taking pre-programmed local responses, these response components generate information describing the threat that was observed and what action was taken to counter it. These responses constitute a tight "inner-loop" of an overall response

ISBN 0-7695-1212-7/01 $10.00 (C) 2001 IEEE

system. By coupling detection and response components into a single response system, an inner-loop responder can implement a response almost immediately after malicious activity is observed. Inner-loop response systems are unable to leverage global state information to improve the quality of response selection.

The information generated by inner-loop response systems can be coupled with system and mission state information to facilitate higher-order decision-making and response selection. The responses resulting from these global decisions are considered "outer-loop" responses. It takes longer to choose a response in the outer-loop since the decision-making engine must communicate with external components to gather the required inputs. However, the advantage of outer-loop response selection systems is that they can make more effective response selections by using data fused from multiple sources. Since outer-loop response systems have a global system view, they are able to pursue a system-wide defense strategy.

Figure 1 shows the relationship between inner and outer-loop responses. Global state is used by an orchestrator to effect coordinated responses, whereas local state is used to directly effect local responses at the behest of simple coordinators.

It is believed that the coordinated response enabled by decision-making in an outer-loop can make better use of currently available response mechanisms, especially when used to defend against determined adversaries or when subjected to diverse attacks. By itself, coordinated response can almost certainly contribute to the success of autonomic defense systems; the availability of greater contextual awareness guarantees that the resulting responses will be as good as, if not better than, decisions based on local information. However, coordinated response is not without cost and risk. It must be determined if the benefits of providing coordination outweigh the cost of gathering and processing the data required to support the coordination functions. In addition, coordinated response systems must resist exploitation by adversaries; no comprehensive techniques for doing this are currently available.

Some prototype systems for providing coordinated response show promise in limited test cases, but they do not scale well enough for actual deployment. Many systems rely on enumerating all possible system state and attacker action combinations and then selecting a response based on the state/action pairs. Even if it were feasible to enumerate all possible system states for every system to be defended, the resulting system would be a brittle hierarchy of "if-then-else" statements that would be unable to accommodate even slight changes to the defended system. Other prototype systems rely on machine learning. A primary goal for these systems is to be flexible enough to defend information systems that change over time. Systems based on machine learning are not sufficient stand-alone solutions in the information assurance domain, although they may be an important component of an overall coordination solution. Attacks typically debilitate a system with such speed and vigor that learning systems do not have the time to adapt to the new threat. If the system is unable to adapt and learn, future occurrences of the same attack will be as successful as the original instance. Post-mortem learning can improve the response to future occurrences of an attack, but it is not an adequate solution since an adversary was allowed to accomplish his objectives. Of course, learning can improve overall system defense if hosts are able to protect themselves using knowledge acquired by observing other hosts that are under attack; it is impossible to provide total system defense, however, based only on learning. Another reason that learning is not ideally suited for information assurance tasks is that learning systems can be exploited by "training" them with increasingly malicious activity. As the activity becomes more malicious, the learning system continues to accept the behavior as benign since it previously observed very similar behavior.

Despite the shortcomings of state/action and learning systems, they make valuable contributions to the overall defense of a system by "raising the bar" for adversaries. They increase attacker workload and may deter some adversaries completely by shifting the risk/reward curve enough so that the attacker pursues other targets. Of course, the benefits of the approaches must be balanced against their weaknesses, such as the possibility that they could unnecessarily deny service to legitimate users.

Since none of the existing approaches to coordination are adequate long-term solutions to the problem, new techniques must be developed. Because of the inherent problems with learning approaches, it appears that a viable solution to the coordination problem must be based on system state and observed adversary behavior. The challenge is to build a system that abstracts nested "if-then-else" logic into a model that scales and adapts to varied systems. Since exact coordination requirements are not known, the SARA architecture is designed to support all forms of coordination.

1.3. A scalable architecture for secure, robust, high-performance communication

Building a system of coordinated components is a challenging task that requires a unifying architecture to provide a common framework in which components can operate. The SARA program is developing an architecture that meets these coordination needs. One of the most important functions of an architecture for coordinated response is providing communication between cooperating components. Communication allows

ISBN 0-7695-1212-7/01 $10.00 (C) 2001 IEEE

components to be integrated with one another to form a cohesive defense system.

Information assurance systems derive efficacy from the ability of their components to communicate with each other or with external entities. For example, traditional network and host-based intrusion detection systems send data gathered at multiple network monitoring stations or hosts to a central administrator's console to facilitate data aggregation and analysis; without such reporting, the benefits of the intrusion detection system would be greatly diminished. Alternatively, directives may need to flow from an administrator's workstation to a firewall in order to change the filtering rules.

Coordinated autonomic response is even more reliant on communication than typical information assurance applications. Most information assurance components will continue to provide fairly good protection to a system even if they are temporarily unable to communicate. In contrast, a coordination engine must have inputs to make an informed response selection, and it must ensure that another component in the system carries out its selected response. Without communication, defense is limited to local decisions and reactions based on limited knowledge since global decisions cannot be made and coordinated responses cannot be taken.

This paper discusses the SARA architecture, which meets the communication needs of autonomic response coordination systems. The SARA architecture also provides a framework for designing and understanding a system protected by coordinated components. The architectural requirements for information assurance systems (with a focus on autonomic information assurance systems) are discussed, and these motivate the presentation of the SARA architecture. Substantial analysis indicates that the SARA architecture will be successful in meetings its goals, but it has not yet been instantiated. A more extensive discussion and analysis of the issues surrounding the development of an autonomic information assurance architecture are being compiled into a comprehensive whitepaper.

2. Architectural requirements

There are many systems available today that, *prima facie*, are able to support communication and coordination in information assurance systems. However, information assurance systems, especially those targeting autonomic response, have demanding requirements that are not fully addressed by the systems available today. There are many systems that address a subset of these issues very well, but none provides a solution that fully addresses them all simultaneously. Furthermore, existing systems do not actively protect themselves from exploitation by an adversary.

The following list is a summary of the most important requirements for an effective architecture that supports autonomic response. The SARA architecture should be:

- **Fast and efficient.** Since reacting to fast attacks (such as a rapidly propagating virus) is a primary goal of autonomic response systems, all system components and communication should be optimized to reduce the latency between the detection of a malicious event and a coordinated response to it. Since attacks may be automated, the system must be able to respond in "computer time". In addition, the use of computer and network resources must be minimized so the mission is not substantially hindered by the defense system.
- **Adaptable and extensible**. The architecture must accommodate networks that change continuously. It must be possible to extend the architecture to new problem areas and to introduce new components without disrupting the existing structure.
- **Introspective.** The architecture must be able to satisfy the information needs of the defensive components. The architecture must provide a system model that describes the physical structure of the network, the processes that are running on the network, and the mission that is being performed. Current and historical information about the entities in the system model must be available to defensive components in a timely manner to facilitate the analysis of attacks.
- **Secure, fault-tolerant, and non-exploitable.** The system must meet the security policies of the network and mission and it must accommodate different security infrastructures. The system must remain robust when faults occur; this applies to random faults caused by the failure of system elements and to malicious faults caused by a deliberate attack on the system. Communication must be reliable so that defensive components remain fully functional even in the face of an attack on the infrastructure. It must not be possible for the attacker to exploit defensive components to effect an undesirable action. For example, the attacker must not be able to trigger a response that causes the system to unnecessarily deny service to legitimate users.
- **Scalable.** The architecture must be able to meet the needs of a network consisting of thousands of machines distributed over a wide geographic area at a reasonable cost. It must also be able to scale down to a single sparsely provisioned host.

Although systems that address these core SARA requirements are available, no single system addresses all of these requirements adequately. Most systems are not intrusion-tolerant; those that are do not offer the low-latency data exchange required to coordinate autonomic response. As a result, a SARA Run–Time Infrastructure (RTI) will need to be developed.

ISBN 0-7695-1212-7/01 $10.00 (C) 2001 IEEE

3. SARA architecture

The SARA architecture consists of a set of components that provide important capabilities to a system and a set of interfaces to those components. The relationships between these interfaces and components are shown in Figure 2.

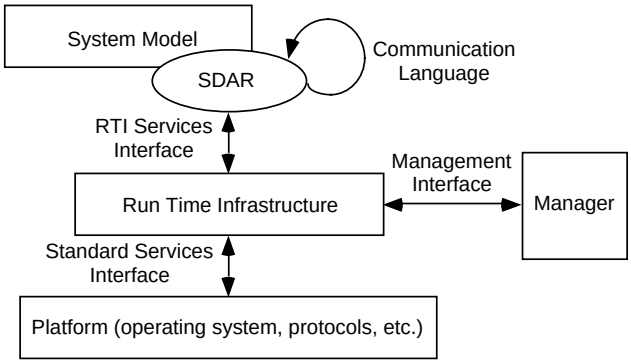

Figure 2 SARA components and interfaces.

The host platform provides standard services to the core of the architecture, the run-time infrastructure (RTI). The RTI provides basic services, such as communication, to defensive components (SDAR components) via the RTI services interface. SDAR components communicate with one another using a communication language that is able to express ideas about the defended system. Information about the defended system is captured in the system model. Managers are able to control the system through the management interface.

3.1. Components

3.1.1. "SDAR" components

SARA relies on cooperating components to provide the information required to coordinate and effect responses. The functions of defensive components can be classified as follows; note that a single software component may assume more than one of these roles:

- **Sensors**. Sensors gather data about the system being defended and make it available to other components. Sensor data consists of reports of events and state. Events correspond to things that have occurred and state corresponds to the status of some aspect of the system. Reports from sensors are considered "unbiased" since they do not contain judgment as to whether the event or state suggests a threat to the system.
- **Detectors**. Detectors determine if sensor output (event and state data) indicates a threat to the defended system by analyzing the sensor output in the context of the system being defended. In other words, detectors turn raw sensor data into information that can be used by decision-making engines. In effect, detectors identify alerts and events from sensors that suggest suspicious or malicious activity and describe the nature of potential threats.
- **Arbitrators**. Arbitrators choose appropriate responses to be taken to maintain the service quality expected of a defended system. Simple arbitrators make pre-programmed, "knee-jerk" responses, but more complex arbitrators use disparate inputs to make reasoned decisions based on an overall system defense strategy. These inputs can include information provided by sensors and detectors as well as system state. Depending on the sophistication of the arbitrator, it may coordinate multiple components to achieve a common goal using explicit directives, or it may orchestrate components by providing a high-level strategy for system defense that components are expected to contribute to.
- **Responders**. Responders are used to effect changes in the system. They act as the actuators in defended systems by taking the actions suggested by arbitrators.

New components developed for use in the SARA architecture will natively participate in a SARA system. Legacy components can participate in a SARA system through the use of adapters that make non–native components function like SDAR components from the point of view of the SARA architecture and other SDAR components.

Since the SARA architecture does not restrict the relationships that SDAR components can have with each other, SDAR components can be arranged to provide the strongest possible defense. Of particular interest is how to organize the arbitrators. The most straightforward approach is for arbitrators to operate independently. Abstraction and scalability could be achieved by using hierarchies of arbitrators. Avoiding centralized coordination by equipping each host with its own arbitrator would provide a high degree of fault tolerance; each arbitrator could make the same decisions since they could perform the same computations on global data.

3.1.2. Run-time infrastructure

The Run–Time Infrastructure (RTI) provides communication and coordination services to SDAR components. As shown in Figure 3, the RTI wraps various standard services such as IP, PKI, encryption, authentication, persistent storage access, and time synchronization in a middleware layer. The RTI provides a high–level service interface tailored to information assurance. This interface defines the functionality of the RTI. The RTI shields its clients from platform differences, allows clients to leverage common services, and compensates for and/or exploits the characteristics of standard services. All inter–SDAR communication should

ISBN 0-7695-1212-7/01 $10.00 (C) 2001 IEEE

occur via the RTI, although some legacy software may need to employ private communication channels.

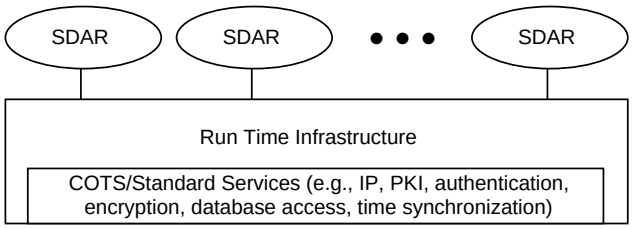

Figure 3 Run-time infrastructure.

The SARA effort will produce an RTI that addresses the requirements presented in section 2 and that can serve as a platform for testing the autonomic response concept. There are many possible RTI implementations that can meet these requirements. Each possible approach to building the necessary infrastructure favors some requirements over the others, requiring tradeoffs to be made. No single RTI will be suited for all possible SARA deployments since network technology and the desired tradeoffs may differ from one deployment to another. The initial SARA RTI targets wired networks and provides low-latency data exchange that is secure, intrusion and fault tolerant, and scalable. To meet these needs, the SARA RTI will use several different design approaches, some of which are discussed here.

- **Distributed caching**. Pertinent data must be cached local to a component since components require access to information even if they become isolated from their peers. Caching data locally also improves information retrieval latencies since most of the information required by components can be retrieved without incurring the overhead of network communication.
- **Widely-disseminated data**. Components that get disconnected from the rest of the SARA system can continue to operate effectively if they have the data that they need in their data cache. Therefore, the probability of a component remaining effective after being disconnected from a network increases as the data cache contains more information. Ideally, components should have access to all of the information exchanged in the system. This allows them to maintain global system awareness and to provide missing information to hosts that may have missed a previous information exchange. Multicast is an efficient way to distribute all exchanged information to all components; it allows clients to share the information they transmit with all of the components in a system with minimal network utilization.
- **Encrypted communication**. To handle the security needs of the system, all information communicated between hosts will be encrypted and authenticated. This provides confidentiality and ensures data integrity. The availability of an infrastructure that allows all SARA components to agree on and share a common symmetric key is assumed.
- **Fixed-size periodic sends**. To avoid traffic analysis attacks and to help the RTI manage network utilization, a fixed amount of data is sent by every component at a regular interval; all inter-component communication occurs through these periodic data exchanges. Since information is sent to all components, an adversary is not able to determine the nature of the data that each component is producing or consuming or which components are cooperating with one another. Communication latency and network resource utilization can be traded off at run-time by changing the amount of data sent and the frequency at which it is sent.
- **Peer-to-peer**. Communication provided by the SARA architecture occurs in a completely distributed manner. Since components communicate directly with their peers, there are no “brokers” or “exploders” to serve as failure points. The communication subsystem has been designed so that an arbitrary number of architecture or defense components can fail and the remaining components will continue to operate to provide the best defense possible.
- **Multi-faceted reliability**. SARA components communicate using UDP. It would be convenient to use TCP to provide reliable communication between components. However, TCP is more vulnerable to denial of service attacks than connectionless protocols, such as UDP. In addition, TCP cannot be used with multicast, which contributes to the overall efficiency of SARA communication. Since UDP is not a reliable protocol, reliability must be provided by other means. One of the techniques used by the SARA architecture is to selectively resend information. On each send cycle, components select data to send to the other components. Some of the data sent repeats previously transmitted data. If another component missed that data when it was originally sent, it will likely receive it on a subsequent transmission. If a component detects that it has missed some data, it can request that other components retransmit that data by issuing a negative acknowledgment for it. The retransmission of data by arbitrary components is possible because of the distributed caching mechanism. To mitigate the effects of dropped or corrupted UDP packets, which could impact an entire information stream, a forward error correction (FEC) protocol will be used. This scheme ensures that even if several packets are lost, the data contained in them can be recreated. This prevents a component from needing to wait for an entire information stream to be retransmitted.

ISBN 0-7695-1212-7/01 $10.00 (C) 2001 IEEE

- **Persistent storage**. Although the distributed caching scheme provides access to most of the previously distributed data, historical data may not be in the data caches; however, historical data might be needed by some SDAR components. To meet this need, a persistent storage mechanism makes all information exchanged in a SARA system available at later points in time.
- **Optimized local communication**. Low-latency communication between components on a single host broadens the responses possible within a high-performance inner-loop. Fast local communication will be achieved by channeling local communication through a high-performance inter-process communication channel.

There are many different RTIs that can be constructed using the concepts in this list. The RTI can be customized based on the characteristics and the requirements of the defended system while providing the same interface to the client components.

3.1.3. System model

The SARA architecture provides a common system model to facilitate effective decision-making. The system model captures the context of a defended system and describes characteristics of the defended system, such as the hardware and software used to support the mission. An important part of the system model is the relationships between components. Relationships represented in the system model might include the network topology, the application programs that are running on the computers in the system, or the current state of the mission. Coordination components can leverage the information in the system model to increase the quality of their decisions. Providing an effective and coherent defense requires that all of the entities in a system share a common system model; if they do not, coordinators will not be able to interpret the state of the system correctly. If different system models are used, the information exchanged between components will be interpreted differently and could result in incompatible actions taking place.

Figure 4 depicts a notional system model. The figure shows three entities (represented as planes) linked by the relationships implied by the arrows. The bottom plane represents the physical hardware in a defended system, such as hosts, networks, and disks. The relationships in the hardware plane represent concepts such as containment, dependence, and connectedness. The middle plane contains the cyber-elements that compose a system. Examples of cyber-elements include processes, files, servers, and SARA components. Relationships between cyber-elements include data flow, containment, trust, and dependence. The top plane represents a mission and contains the applications being defended. The mission is described in terms of plans, tasks, goals, and related concepts.

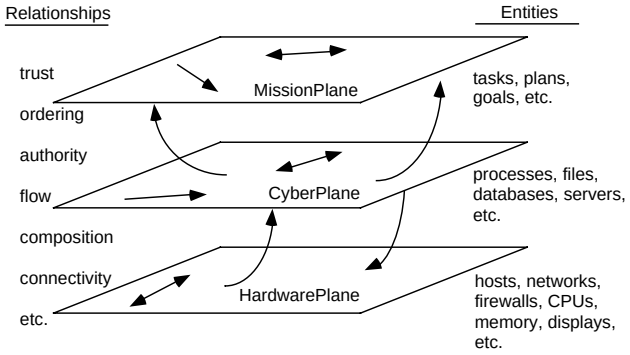

Figure 4 Notional system model.

A system model is composed of a system model template and a system model instance. The system model template consists of class definitions for the entities and relationships captured by the model; it is a superset of all of the systems SARA is capable of defending. The system model instance uses instantiations of the classes defined by the system model template to represent the actual system being defended.

The system model is an important but complex component of the architecture. Efforts such as CIM [2], CCS [1], IDMEF [4], and CISL [5] are applicable to this problem, and the work done by these projects will be leveraged when designing the system model used by SARA systems.

3.1.4. Managers

Managers control and monitor SARA systems. They provide functions such as status monitoring, status reporting, configuration, operator displays, debugging services, initialization, finalization, and performance tuning. These functions help human operators monitor, configure, and make suggestions to a SARA system.

A "host manager" is a good example of a SARA management component. A host manager will execute on each host to coordinate component startup and shutdown, status reporting, and related functions. It will also ensure the continued availability of defensive components by maintaining their integrity and restarting failed components.

3.2. Interfaces

The SARA architecture provides four interfaces that can be used to interact with the SARA infrastructure, SDAR components, and the defended system. The communication language and RTI services interface are

ISBN 0-7695-1212-7/01 $10.00 (C) 2001 IEEE

used by SDAR component developers to interface with other SDAR components and the SARA system. The RTI leverages functionality provided by the operating system and other COTS/GOTS systems through a standard services interface. SARA system administrators are able to monitor and control SARA systems through a management interface.

3.2.1. Communication language

SARA requires a communication language that defines the information that SDAR components can communicate with one another. Although the communication language does not map directly to the system model, it does provide facilities that allow SDAR components to communicate about many system model objects. It must also allow SDAR components to communicate information that is not captured in the system model, such as alerts.

Although the SARA architecture is flexible enough to accommodate a wide variety of languages, the selected language will be based on a class hierarchy. Representing the language as a class hierarchy allows information to be communicated as objects and helps organize information so that consumers can use it more efficiently. The use of a hierarchy also promotes extensibility, since new message subtypes can be added without modifying existing SDAR components. The top levels of a possible inheritance hierarchy for a communication language are shown in Figure 5.

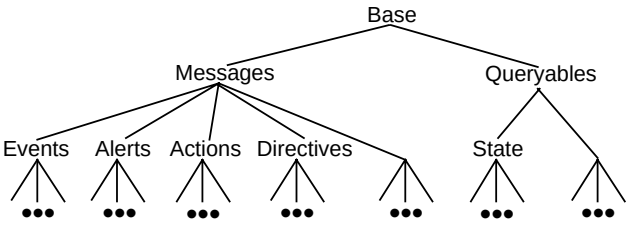

Figure 5 Communication language class hierarchy.

The need to balance low data exchange latency with low resource utilization suggests that partitioning information to be communicated into two categories may be useful. Indeed, the SARA architecture uses two distinct information types: messages and queryables. The primary distinction between messages and queryables is how data is communicated among components. Messages are sent at the discretion of producers: they are pushed. Queryables, in contrast, are sent at the request of a consumer: they are pulled. Messages provide low data exchange latency since information is sent to consumers as soon as it is available. Information expressed as a queryable must be requested by a consumer. Since components request queryable information only when it is needed, communication and processing resources are not consumed if the data is not required by any components. Since consumers do not know when queryable information will become available, they must poll for it; polling introduces some latency into the information exchange process. Because of this latency, queryables are not generally suited for communicating information that directly invokes a response. Rather, queryables should typically be used to communicate information that assists in decision making, such as system state.

There is a great deal of information that needs to be conveyed using the language. Components must be able to communicate about events that they have observed and the state of the system. In particular, they must be able to rapidly communicate alerts containing information about suspicious activity that has been detected in the defended system. Decision-making components must be able to issue directives that cause response components to take some action. Finally, components must be able to acknowledge that they have taken an action in response to a directive. There may be information that components need to communicate in addition to the items mentioned here. The architecture does not constrain the information that can be conveyed by the language.

The characteristics of the language used by SDAR components influence the amount and type of knowledge that must be distributed in a SARA system. For example, the language could communicate very concrete directives that request certain components to take specific actions, or the directives could be abstract requests that components convert into concrete actions that they can carry out. The concrete directive approach requires the coordinator to understand the capabilities of the SDAR components in the system and it tightly couples coordinators and the available responders. The abstract directive approach does not require the coordinator to have very much knowledge, but it requires response components to have situational awareness of the system so that they can determine how to effect the necessary changes without adversely impacting other components. The architecture is capable of supporting both types of directives equally well.

Some portions of the language space have been addressed by programs such as IDMEF [4], CISL [5], and CCS [1]. SARA will use the output of those efforts as the basis of its communication language. These language efforts are incomplete, however, and do not address some of the information that will need to be communicated by SDAR components. Notably, the SARA project will need to focus on how to express responses in the context of existing languages.

3.2.2. RTI services interface (API)

The SARA API abstracts the RTI services into a collection of classes that give the component developer the power and flexibility to develop robust and powerful

ISBN 0-7695-1212-7/01 $10.00 (C) 2001 IEEE

SDAR components independent of the supporting infrastructure. The API provides very general support for communication between SDAR components and basic services that are relevant to all SDAR components. While many API services are provided explicitly through methods in the client base class, other important functionality is provided implicitly.

The API decouples the interface that component developers use to build SDAR components from the implementation of that functionality. This allows components to remain static even as the SARA system evolves. The API constrains clients to defined behaviors and semantics, which helps the system meet the requirements presented in section 2.

The services explicitly provided by the API fall into the following categories:

- **Creating and destroying SDAR components**. Since the SARA API is provided as a collection of classes, objects corresponding to each SDAR component must be created, and then later destroyed. This creation/destruction process affects only the component's interface to the RTI; the component itself is not affected.
- **Joining and leaving SARA systems**. An SDAR component must authenticate to and join a SARA system before it can communicate with other SDAR components. Once a component joins a system, it agrees to participate in the system's defense.
- **Messaging**. The API allows components to push messages to other components. These messages are communicated as objects since objects are efficient and easy to deal with. Messages are not addressed to specific components. Instead, a publish and subscribe system is used. In a publish and subscribe system, messages are implicitly addressed; the sender does not need to know the recipients of a message since the RTI routes messages to the clients that previously requested them. A publish and subscribe scheme decouples information producers and consumers and reduces the chance of real or artificial dependencies being created since there is no mechanism that clients can use to communicate with specific components. This increases system survivability since the failure of a single component will not directly impact other components in the system. Decoupling components also creates opportunities for components to collaborate with other components to achieve common goals. A publish and subscribe scheme is simple for clients to use and amenable to very efficient implementation. Clients are able to express the data that they will send and receive using a message template system that allows clients to precisely specify the messages that they are interested in. When a component receives a message that it has subscribed to, the RTI signals the component to handle the message.
- **Querying**. Components are able to pull system state information from other components that have agreed to provide the required information; this is accomplished by issuing a query. Queries consist of an object to be filled in with information and an identifier that denotes the system object that the information is needed about. Once a query has been created, the information object can be updated at the request of the client. A default "root" object is provided to allow components to completely discover the entities in the system it is participating in without any prior knowledge of the system. Queries are answered by components that previously stated that they are able to do so.
- **Client/RTI communication**. To ensure that SDAR components are serviced in an efficient manner, information must flow between components and the RTI. The RTI may ask clients to adjust their operating parameters, or a component may report its current state or intent to the RTI so that the RTI can plan to accommodate future client actions. The API facilitates this information exchange in three ways: clients can explicitly pass information to the RTI; the RTI can make explicit requests of clients; and the RTI can observe client behavior through the client's use of the API.

The capabilities provided by the API will be adequate for most SDAR components, but some components will inevitably have specialized needs that are not met by the API. Higher–level functionality can be composed from the building blocks provided by the RTI. In addition, most API functionality can be modified by overriding the default implementation of API calls or extending the default interface provided by the API.

3.2.3. Standard services interface

The standard services interface provides the RTI with access to standard services. Examples of standard services include services provided by modern operating systems (e.g., filesystems, devices, communication, threading, IPC), security (e.g., PKI, encryption, authentication), and COTS/open source libraries. The specific services relied on by any given implementation of the RTI will vary, and are therefore not detailed as part of the architecture.

3.2.4. Management interface

A management interface supporting the required monitoring and control functions will be provided. These functions help human operators monitor, configure, and make suggestions to a SARA system. Some of these functions include monitoring and reporting system or

ISBN 0-7695-1212-7/01 $10.00 (C) 2001 IEEE

component status, allowing an operator to configure a system or view its current configuration, providing debugging information to component developers, and gathering performance data and allowing performance-related parameters to be tuned.

The specific requirements for this interface will be determined through experimentation and further analysis and may be tailored to specific SARA system deployments.

3.3. SARA component organization

To this point, the SARA system has been described as a collection of individual components and interfaces; however, it is advantageous to organize the components of the system into a hierarchy. The SDAR components at any given level in the hierarchy are said to be a member of the same cell. There are three advantages to arranging components into hierarchical cells: it allows SARA to be scaled to very large networks; it provides convenient boundaries at which the information flow can be managed so that only relevant information is transmitted to SDAR components; and it provides a framework for resolving inter-domain incompatibilities in technology, policy, or requirements.

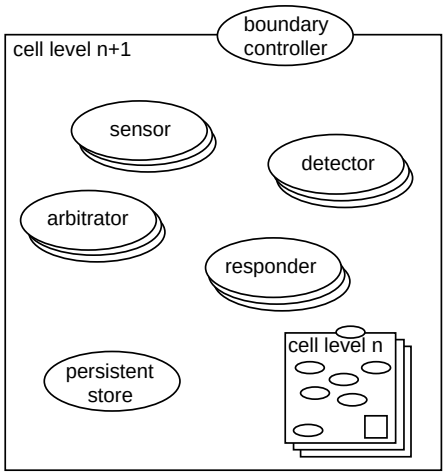

Figure 6 A sample cell.

The cell is the fundamental unit of abstraction used to organize SDAR components. As shown in Figure 6, a cell may contain SDAR components, persistent storage, a cell boundary controller, and other cells. The Level 0 cell, which is usually the lowest-level cell, includes the SDAR components associated with a single host computer. Level 0 cells are grouped into Level 1 cells, Level 1 cells are grouped into Level 2 cells, and so on; there is no theoretical limit to how deeply cells can be nested, nor is there any definition of what cell levels other than Level 0 correspond to. If the RTI described in this document is used, a cell would also be a multicast group, an encryption domain, and a common security policy domain. Cells at the same level do not overlap; components must be a member of exactly one cell. Cell boundaries do not have to correspond to any physical boundaries, such as subnets, but it is often useful to organize cells along such boundaries.

The cell boundary controller (CBC), which is part of the RTI, is a cell member that acts as an intermediary for all communication between the components of the CBC's cell and the next higher-level cell. Since the CBC logically sits at the boundary between two cells, it is ideally situated to filter, route, aggregate, translate, and proxy information. As an example of how a CBC mediates between cells, consider a Level 0 cell (corresponding to the SDAR components associated with a host) that is a member of a Level 1 cell. In the Level 0 cell, all communication between SDAR components is through shared memory without encryption since a host is assumed to be secure and reliable. However, messages that should be passed to the Level 1 cell must be forwarded by the CBC. The CBC will ensure that the information is communicated using a secure and reliable transmission protocol. Similarly, the CBC for the Level 1 cell relays information, as needed, to the Level 2 cell. This will generally involve a change of multicast group and encryption keys.

To promote scalability, the amount of information flowing into higher-level cells must be moderated. This can be done by aggregating information and restricting the information that passes through the CBC to the information that contributes to defense and decision making in the higher-level cell. This information flow is consistent with arranging the SARA coordinators in this same hierarchy, thus having the control flow follow the main channels of the communication flow.

The SARA architecture can be adapted to heterogeneous networks with different domains. For example, one part of a defended system may employ different technologies (e.g., wired or wireless) or impose different requirements (e.g., mission, security policy, administrative control) than other parts. The cell structure provides a convenient way to deal with the requirements of the different domains in a defended system. The deployment and organization of the SDAR components in any cell may differ among domains of the defended system, but each cell provides a standard interface to the cells above and below it, even if those cells differ in internal organization. By aligning cell boundaries with administrative and policy boundaries, the system designer can devise and implement the appropriate algorithms for the cells of each domain and customize the communication between domains.

Certain components, such as persistent storage controllers, may serve many cells. It is most convenient to locate these components in a cell that spans the entire

ISBN 0-7695-1212-7/01 $10.00 (C) 2001 IEEE

domain of the component. However, there are other ways of handling these special cases as well.

4. Proof of concept experiments

With a SARA architecture researched and designed, the SARA project must focus on demonstrating that the architectural principles in this document are sound and can be adapted and extended so that field–deployable SARA systems can be built. In 2001, the SARA architecture will be partially instantiated and used in a series of "proof of concept" experiments to demonstrate the validity of the SARA approach and to serve as the basis for further autonomic defense research. To demonstrate the capabilities of the SARA architecture and the autonomic defense approach, the experiments will require an RTI, a language, SDAR components, and scenarios. The core pieces of the architecture that need to be tested in the POC experiments are:

- **Intra–host communication** based on an efficient IPC mechanism that communicates information using objects.
- **Inter–host communication** that communicates data among hosts as objects and is based on UDP multicast and a reliability mechanism that uses periodic transmission, forward error correction, and selective retransmission.
- **Partitioning** that organizes SARA system components into cells to facilitate scaling, resolution of inter-domain issues, and overlaying SARA on a system to be defended.
- **Messaging** that uses a publish and subscribe system to distribute data to cell members and that can selectively forward messages across cells.
- **Querying** that distributes queries to all components in a cell and routes an appropriate response back to the querying component.
- **Security** that provides protection against traffic analysis and can integrate with an external COTS or GOTS system to provide confidentiality and authentication.
- **Persistent storage** that facilitates access to historical SARA data.

A set of well-defined experiments that show the true value of coordinated autonomic response in a realistic environment will test and demonstrate the SARA concept. Some of these experiments must be based on scenarios that demonstrate the ability of a coordinated system relying on the SARA architecture to provide a stronger defense than is possible without relying on global coordination. In addition to testing the SARA architecture using scenarios, live red-team experiments should be conducted to verify the functionality of the RTI.

5. Summary and next steps

Coordinating autonomic detection and response systems is a promising approach for improving the quality of defense provided to information systems. An architecture is one of the two important elements required to build a system that provides coordination. The SARA architecture meets the communication requirements of systems that facilitate coordination among separate components working to defend a single information system; it can also be used to facilitate robust communication for systems outside of the information assurance domain. The SARA architecture defines the roles of components that can sense, detect, arbitrate, and respond to suspicious activity and it provides a run-time infrastructure that enables these SDAR components to interoperate. A communication language and system model help components communicate with one another and reason about the current state of the system. A simple API allows components to harness the functionality provided by the RTI. Components are organized into cells to meet administrative and functional requirements and to allow SARA to be scaled to defend very large systems. The SARA concept and architecture will be tested in a series of "proof of concept" experiments scheduled to occur in 2001. These experiments will use an RTI implementation based on the principles outlined in this document. The results of the experiments will shape future versions of the RTI, but the API that clients use to participate in the system will remain constant.

Future architecture research will be based on the outcome of the proof of concept experiments and will likely focus on increasing the ability of the architecture to adapt itself to the system it is defending. Other important areas of research include designing systems to facilitate robust communication in resource constrained environments, developing a standard language that SDAR components can use to communicate, defining the contents of an effective system model, developing response techniques suited for autonomic defense, and defining a model for scalable and adaptive response selection and coordination.

6. Acknowledgments

This work has been shaped by many of the ideas and concepts developed by the SARA Working Group. The following organizations contributed to this work through their involvement in the SARA Working Group: BBN Technologies, DARPA, Integrated Management Services, Inc., Sandia National Laboratory, SPAWAR, Syntek, and Telcordia Technologies.

ISBN 0-7695-1212-7/01 $10.00 (C) 2001 IEEE

7. References

[1] Information on CCS may be found at the 3GS web site at https://archive.ia.isotic.org/.
[2] The CIM specification and related documents and tutorials may be found at the Distributed Management Task Force web site http://www.dmtf.org/.
[3] Cyber Panel Mission Model Working Group, "Strawman Mission Model Version 1.0," 4 December 2000.
[4] Internet drafts and related information on IDMEF may be found at IETF web site at http://www.ietf.org/html.charters/idwg-charter.html.
[5] R. Feiertag, C. Kahn, P. Porras, D. Schnackenberg, S. Staniford–Chen, and B. Tung, "A Common Intrusion Specification Language (CISL)," 11 June 1999.

ISBN 0-7695-1212-7/01 $10.00 (C) 2001 IEEE