A Security Architectural Approach for Risk Assessment Using Multi-agent Systems Engineering

Gustavo A. Santana Torrellas

Instituto Mexicano del Petróleo
Programa de Matemáticas Aplicadas y Computación
Eje Central Lázaro Cárdenas N°152
CP 07730, México, D.F.
gasantan@imp.mx

Abstract. The analysis of incidents resulting in damage to information systems show that most losses were still due to errors or omissions by authorized users, actions of disgruntled employees, and an increase in external penetrations of systems by outsiders. Ideally, information systems security enables management to have confidence that their computational systems will provide the information requested and expected, while denying accessibility to those who have no right to it. Traditional controls are normally inadequate in previous mentioned cases or are focused on the wrong threat, resulting in the exposure of vulnerability. Security is a critical parameter for the expansion and wide usage of agent technology. A threat model is constructed and subsequently the basic techniques to deal effectively with these threats are analyzed. Then this paper presents a dynamic, extensible, configurable and interoperable security architecture for multi-agent systems applied to security assessment services. It is explained how this architecture can be used to tackle a big part of security threats. All the components of the security architecture are analyzed while we also argue for the benefits they offer. . Such information security changes often encourage the creation of new security schemas or security improvements. Accommodating frequent systems information changes requires a network security system be more flexible than currently prevalent systems. Consequently, there has recently been an increasing interest in flexible network security and disaster recovery systems.

1 Introduction

As the complexity of today's distributed computing environments continues to evolve independently, with respect to geographical and technological barriers, the demand for a dynamic, synergistically integrated, and comprehensive information systems security control methodology increases. Unfortunately, the prevalent attitude toward security by management and even some security personnel is that the confidentiality of data is still the primary security issue. That is, physical isolation, access control, audit, and sometimes encryption are the security tools most needed. The primary goal of any enterprise-wide security program is to support user communities by providing cost-effective protection to information system resources at appropriate levels of integrity, availability, and confidentiality without impacting security, innovation, and creativity in advancing technology within the corporation's overall objectives. Ideally, information systems security enables management to have confidence that their computational systems will provide the information requested and expected, while denying accessibility to those who have no right to it. People dealing with security have a hands-on experience with such issues. A secure system is a system that provides a number of services to a selected group of users and restricts the ways those services can be used. A security service is a software or hardware layer that exports a safe interface out of an unprotected and possibly dangerous primitive service. In order to build a security service we need a security architecture. Having analyzed the security needs of the Mobile Agent (MA) technology we propose in this paper a dynamic, extensible, configurable and interoperable security architecture for mobile agent systems. Software agents [1] are a rapidly multi-directional developing area of research since the early 90s. Mobile Agents shatter the notion of client/server model and eliminate its limitations. Standardization efforts and guidelines that boost the usage of agent technology exist in organizations such as the Object Management Group [11] and the Foundation for Intelligent Physical Agents [12]. Agents are computer and transport independent (they depend only on the execution environment) and therefore promote interoperability among systems and software. Integrity and availability must be addressed as well as ensuring that the total security capability keeps current with technology advancements that make it easier to share geographically distributed computing resources. Security environments have introduced significant opportunity for process reengineering, interdisciplinary synergism, increased security, profitability, and continuous improvement. Enterprise-wide security programs, therefore, must be integrated into a systems integrity engineering discipline carried out at each level of the organization and permeated throughout the organization.

2 Threats in a Distributed Agent Environment: Understanding Distributed Processing Concepts and Corresponding Security-Relevant Issues

There are so many factors influencing security in today's complex computing environments that a structured approach to managing information resources and associated risk(s) is essential. New requirements for using distributed processing capabilities introduces the need to change the way integrity, reliability, and security are applied across diverse, cooperative information systems environments. The formal process for managing security must be linked intrinsically to the existing processes for designing, delivering, operating, and modifying systems to achieve this objective. The operational environment for distributed systems is a combination of multiple separate environments that may individually or collectively store and process information. The controls over each operational environment must be based on a common integrated set of security controls that constitute the foundation for overall information security of the distributed systems. Distributed systems are an organized collection of programs, data, and processes implemented in software, firmware, or hardware that are specifically designed to integrate separate operational systems into a single, logical information system infrastructure.

The foundation of security-relevant requirements for distributed systems is derived from the requirements specified in the following areas:

- Operating systems and support software,
- Information access control,
- Application software development and maintenance,
- Application controls and security,
- Telecommunications,
- Satisfaction of the need for cost-effective security objectives.

Mobile code programming is by its nature a security-critical activity. In an agent based infrastructure the security implications are far more complex than in current static environments. In such an environment author of the MA code, the user, the owner of the hardware, the owner of the execution platform (even the execution place) can be different entities governed by different security policies and possibly competitive interests. In such a heterogeneous environment security becomes an extremely sensitive issue. We identify the threats that exist in an agent-based infrastructure. We can have: misuse of execution environment by mobile agents, misuse of agents by other agents, misuse of agents by the execution environment, misuse by the underlying network infrastructure.

Our approach also provides protection for the two first categories and tries to provide some guarantees to the agent concerning the host code and execution environments.

3 Dealing with Security Risks

Having presented the threat model we will try here to see how we can deal with these problems. Traditionally, when talking about data security usually three security objectives are identified: confidentiality, integrity, and availability. To better suit the needs of electronic security management with all its legal aspects more security objectives have been identified. The most important one is accountability. In such a way the four main security requirements to be satisfied are:

- *Confidentiality.* Describes the state in which data is protected from unauthorized disclosure. A loss of confidentiality occurs when the contents of a communication or a file are disclosed. Private data carried by the agent or used by the platform (such as audit logs) should remain private. Intra- and inter- platform communication should by no mean be revealed to 3rd parties by monitoring or other techniques.
- *Integrity.* Means that the data has not been altered or destroyed which can be done accidentally (e.g. transmission errors) or with malicious intent (e.g. sabotage).Agent code should be protected from unauthorized or accidental modification of code, state and data.

- *Accountability*. If the accountability of a system is guaranteed, the participants of a communication activity can be sure that their communication partner is the one he or she claims to be. So the communication partners can be held accountable for their actions. Agents and platforms should audit their activities and be able to provide detailed info for debugging or security purposes. Every action should be uniquely identified, authenticated and audited.
- *Availability.* Refers to the fact that data and systems can be accessed by authorized persons within an appropriate period of time. Reasons for loss of availability may be attacks or instabilities of the system. Resource management, controlled concurrency, deadlock management, multi-access, detection and recovery from faulty states such as software and hardware failures apply to mostly to platforms.

Several approaches have been developed in order to minimize security risks.

4 The Security Risk Analysis Process

The methodology of security risk analysis also comprises a number of basic steps. These differ between authors, but in general include:

- *Asset Identification*. The asset identification phase should identify the resources that require protection. These will include: hardware, software, data, documentation, and computer services and processes. Financial values can be readily applied to some of these assets, but others are more difficult to price.
- *Vulnerability Analysis*. Having listed the assets of a computer system, the next stage is to determine their vulnerabilities. This stage is more difficult than the first, as it requires a degree of imagination to predict what damage might occur to the assets and from what sources [2]. The general aims of computer security are to ensure data secrecy, data integrity and availability. System vulnerabilities are situations that could cause the loss of any of these qualities. A thorough understanding of the threats to the system is required if all the vulnerabilities are to be identified. Methodical and structured approaches are required if threat identification and vulnerability analysis is to be successful.
- *Likelihood Analysis*. The aim of likelihood analysis is to ascertain how often the system will be exposed to each of the vulnerabilities identified. Likelihood relates to the current security safeguards and the environment in which they are applied. Estimating the probability of exposure to a threat can be difficult. Sources of data for this estimation include: operations logs, local crime statistics and user complaints.
- *Countermeasure Evaluation*. All the analysis so far reflects the current situation. If, from this analysis, it is determined that the projected loss will be unacceptable, new or alternative countermeasures will have to be investigated. New controls will have to be identified, and their effectiveness evaluated.

5 Credentials and Authentication

Because agents are programs, they are intangible and live in a virtual world, we connect the trust model of such an infrastructure with the trust model of real world in order to make security critical decisions. That basically means that since every agent acts on behalf of a user or generally an entity we check to see if we trust that entity and indirectly trust the agent. An agent is signed by one or more entities. Those entities can be either the creator of the code, the user that dispatched the agent (usually this is also the creator), a place of a host and generally any entity that holds a valid certificate. Signing an agent guarantees that i) the creator is the one claimed by the agent, ii) agent's code (at least the signed part) has not been tampered by a 3rd party during transportation. Signing doesn’t guarantee that the agent will execute correctly (safety). Furthermore one place can encrypt the agent with the public key of the destination place (only the destination place has the private key to decrypt the agent), protecting in this way the agent while it traverses the net until it reaches the final destination. TLS standard (Transport Layer Security) [5] is also another option. Credentials also touch indirectly the “malicious host" problem. Since each place (or at least each agency) has its own certificate there is proof that this agent is mapped to a legal user who bears responsibility of the behaviour of the agency. Non-changing parts of the agent should be signed for maximum protection.

5.1 Means of Authentication and Authorization

In this section the actual methods or processes that are used to authenticate the identities of users are discussed. The authorisation of the user to gain access to services or resources can be carried out in a system after the user has been authenticated and his identity is resolved through the use of access control lists (ACL), to determine what the particular user is authorised to do. Authorisation is thus at maximum as accurate and correct as the process of authentication. A mechanism like the SPKI could be used, to avoid authentication of the user, but to still provide a reliable authorisation.
Some mechanism for implementing mobile authentication and authorization are:

- **Passwords –** Passwords associated to user names (something that the person knows) are a simple way of authentication. There are several authentication schemes that make use of passwords in combination with some other factor. A simple extension of passwords is one-time passwords.
- **Password with a token –** Passwords can be used in combination with some physical object (something that the person owns). This concept has been extended with the use of integrated circuit cards (ICC) or smart card. A challenge and response method is used in authenticating the user. ’Synchronous one-time passwords’ [5] is another similar technique.
- **Biometrics –** Biometrics authentication techniques include fingerprint recognition, retinal scanning, hand geometry scanning and handwriting and voice recognition [5]. These techniques are all based on the physical properties of a person (something he owns / is).
- **Digital Signatures –** When a PKI is put in place, digital signatures can be used to authenticate users. The following sequence of actions has to be carried out in order to authenticate a user by his digital signature:

1. The user requests access to the service or system
2. The system generates some data for the user to encrypt using his private key. Then the data is sent over to the user.
3. The user concatenates the data received from the system and a time stamp and encrypts the whole sequence. (N.B. It is a good practice that e.g. a time stamp is concatenated to the data, so that the data to be encrypted cannot completely be decided by some untrusted party. This is to avoid the possibility of a 'Chosen plain-text attack' as described in [12].) Then the encrypted data (the cipher text) is sent back to the system. Along with the encrypted data a link to the certificate (or the certificate itself) of the user is sent.
4. The system decrypts the received information with the public key of the user, found in the certificate.

The system verifies that the decrypted information is composed of the originally generated data and a valid timestamp. If this seems to be OK, the system has successfully authenticated the user.

5.2 Properties of a Good Authentication and Authorization Mechanism

This section lists the properties that a good identity authentication and authorization mechanism should possess. Some of the features listed are in contradiction of each other, but mostly it should be possible to reach an acceptable Level of compliance with each of the criteria:

- **Correctness –** The results of each individual instance of authentication or authorization carried out should be correct. If it is possible to authenticate the user, the result should always be that either it is found, that the user is who he claims or he is found to be a fraud. Based on this perfectly correct authentication it is further possible to authorize the user to access those services and resources that defined to be accessible for him or to the group or groups he belongs to. In practice it is impossible to get an absolute certainty in authentication. Only a reasonable Level of certainty can be gained.
- **Possibility to anonymity and privacy –** Identity authentication should only be done when absolutely necessary. Whenever authorisation is possible without the user's identity being revealed, it should be done that way.
- **Speed –** The process of authentication should be fast. The user shouldn't have to wait for the result for more than a second does or two.
- **Attack resistance –** The perfect mechanism of authentication should be resistant against any known or unknown types of attacks.
- **Inexpensiveness –** The mechanism shouldn't require extensive investments from either the users or the authenticators.
- **User friendliness –** The mechanism should produce as little overhead to the user as possible. It should also be as easy to use and understand as possible. In the optimum situation the user doesn't have to perform any actions in order to become authenticated. The user shouldn't be forced to carry around any extra equipment, magnetic or smart cards, lists of passwords or other physical objects in order to use the system.

- **Universality –** It should be possible for the user to use the same means or method of authentication in all services and everywhere**.**

5.3 Access Control Checks

Having successfully identified the agent is only the first step. Trust in the agent's credentials doesn't guarantee that it will behave legitimate nor execute correctly. Thus we monitor and authorize every call it makes to platform's resources. Any access to any resource e.g. network, file, system configuration etc is subject to an access control check. Therefore we need a policy and an enforcement manager to make sure that our policy is enforced. With this second level of check we provide fine-grained control customized per user or group. As users perform various activities not all of them have the same rights. The security is based in protection domains of Java. Those protection domains are defined by the internal agent id (not immutable) and/or by the signer(s) of the agent code (immutable). We can even require a combination of user identities in order to allow an agent to perform a task. A flexible policy scheme guarantees exactly that. Although this second level provides some extra and selective security we understand its limits.

6 The Security Architecture

Secure systems and security applications produce their own special challenges to usability. Since secure systems have been traditionally difficult to use, some people are under the impression that usability and security are innately in conflict. Our model will have to provide a powerful tool for the definition of security policies, but power (that is expressivity) is useless if the user cannot easily figure out how he can employ it. For those reasons, the first requirement of our model is to provide a Graphical User Interface (GUI) that hides the complexity to the user. The GUI should provide the user with:

1 a way to define and modify the security policy
2 a tool to check the policy behaviour
3 a help to "debug" the security policy

Because information technology security planning is primarily a risk management issue, the architecture model, his policy and its associated standards and guidelines focus on the creation of a shared and trusted environment, with particular attention to:

- Common approaches to end-user authentication;
- Consistent and adequate network, server, and data management;
- Appropriate uses of secure network connections;

Approaches that try to incorporate security after the design phase have been proven to fail. The security architecture (Figure 2) for mobile agent systems tries to incorporate all above solutions to the threat model presented before and also to be as open as possible in order to integrate easily future solutions. Furthermore we follow in this approach the MASIF standard for interoperability reasons. The main requirements of the implementation are:

- be secure; for that purpose a general architecture will be defined. Its weak points will have to be detected and protected;
- provide an efficient way to evaluate policies;.

6.1 Places

The agent system (Figure 1) consists of places. A place is a context within an agent system in which an agent is executed. This context can provide services/functions such as access to local resources etc. A place is associated with a location which consists of a place name and the address of the agent system within which the place resides. Places can contain other places. Places are i) dynamically assigned to agents as they enter the agency based on some criteria e.g. all agents coming from a specific user or location or agents belonging to a specific policy scheme etc. or ii) statically (permanently) assigned per entity (e.g. User, enterprise etc). In the latter static resources are given to the place (after agreement with the node provider) and the local resource manager manages them. This offers several advantages e.g. secure communication or paths between organisation-trusted agents etc.

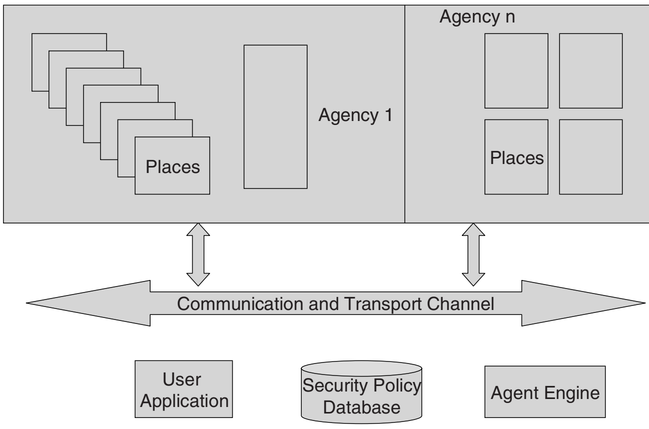

Fig. 1. The Distributed Agent Environment

A policy scheme and a resource access scheme are assigned to each place and the respective policy and resource manager are given the general security guidelines, which can never be bypassed. If an agent has sufficient credentials, then it can fully interact with the components e.g. change the place's policy, ask for more resources, insert elements in the component database etc. Of course advanced security facilities offered by the place can be used to minimize these risks (e.g. a secure communication service via the platform). Furthermore if one wants can use a place as a Test Place (a firewall like approach) and allow suspicious agents to execute there, monitor the results and then determine if it will allow them to execute in the real place. Certainly if you see for instance that an agent changes inappropriately the policy file of the Test Place you forbid it to execute into the desired place. Also agents are somehow isolated since each one has its own class loader. Places beyond having unique IDs, also hold their own public/private keys. An agent can ask to be signed in order to have a proof that it passed via this place. This also helps with the so-called "multi-hop" security problem. If every place signs a specific part of the agent then we can trace back the exact route the agent followed. Based on that info we can take further security decisions.

6.2 Policy Manager

The Policy Manager is responsible for managing the policy schemes stored in the policy database. By separating the policy DB from the enforcement engine we insert a dynamic way of policy modification. The security policy defines the access each piece of code has to resources. Signed code can run with different privileges based on the identity of the person or place who signed it. Thus users can tune their trade-off between security and functionality (of course within limits given by administrator).

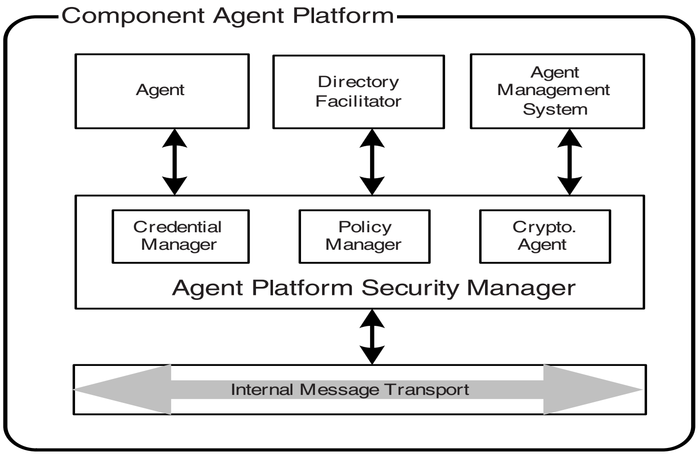

Fig. 2. Secure Agent Platform Architecture

When an agent comes to an agency then he is subjected first to the general agency's policy which is set by the user that initiated the agency (Figure 3) and is considered to be the super-user. Subsequently after passing successfully that control the agent is subjected to the place's specific policy. It is clear that with this sequential check of policies we avoid the problem of granting contradictory access rights for the same action by different policies. One can also simply forbid agents from a specific user/domain for personal. Any attempts to describe the security policy in terms of each individual principal's authority to access each individual object is not scalable and not understandable for those instituting the policy. Making security decisions rather than the individual identities. So we have role-based policy, group policy, clearance labels, domains etc. Furthermore by grouping policies we allow for faster execution times while trying to enforce the policy. In our system all security checks are identity-based in order for an agent to enter a place. After an agent successfully enters a place future security checks become role-based. Thus we don't have each time to verify agent's credentials. We check only to see in which place the agent resides and what the appropriate policy for that place is.

This approach is once more followed in our effort to speed up security checks and improve architecture's performance.

6.2.1 General Security Policy

It is the IT security policy of the policy manager that:

1. Each agency shall operate in a manner consistent with the maintenance of a shared, trusted environment within resource and component manager for the protection of sensitive data and security transactions. Agencies may establish certain autonomous applications, including those hosted by an Applications Service Provider or other third party, outside of the shared, trusted environment, PROVIDED the establishment and operation of such applications follows all guidelines as set forth in this security policy and does not jeopardize the enterprise security environment, specifically:
   - The security protocols (including means of authentication and authorization) relied upon by others; and
   - The integrity, reliability and predictability of the Organisational backbone network.
2. Each agency shall establish its secure state security applications within the guidelines of the Policy Manager and resource manager Network Infrastructure. This requires that all parties interact with agencies through a common security architecture and authentication process. Enforcement Engine shall maintain and operate the shared infrastructure necessary to support applications and data within a trusted environment.
3. Furthermore, each agency that operates its applications and networks within the whole Organisational Network Infrastructure must subscribe to the following principles of shared security:
   - Agencies shall follow security standards established for selecting appropriate assurance levels for specific application or data access and implement the protections and controls specified by the appropriate assurance levels;
   - Agencies shall recognize and support the state's standard means of authenticating external parties needing access to sensitive information and applications;
   - Agencies shall follow security standards established for securing servers and data associated with the secure application; and
   - Agencies shall follow security standards established for creating secure sessions for
   - Application access.

4. Each agency must address the effect of using the Internetworking protocols to conduct transactions for state security with others. Plans for Internet-based transactional applications, including but not limited to e-commerce, must be prepared and incorporated into the agency's portfolio and submitted for security validation.
5. Each agency must review its Information transactions security processes, procedures, and practices at least annually and make appropriate updates after any significant change to its security, computing, or telecommunications environment. Examples of these changes include modifications to physical facility, computer hardware or software, telecommunications hardware or software, telecommunications networks, application systems, organization, or budget. Practices will include appropriate mechanisms for receiving, documenting, and responding to security issues identified by third parties.
6. Each agency must conduct an IT Security Policy and Standards Compliance Audit once as frequently as possible. The audit must be performed by knowledgeable parties independent of the agency's IT organization, such as the General Agent Auditor. The work shall follow audit standards developed and published by the General Agent Auditor. The General Agent Auditor may determine an earlier audit of an agency's IT processing is warranted, in which case they will proceed under their existing authority. The nature and scope of the audit must be commensurate with the extent of the agency's dependence on secure IT to accomplish its critical security functions. Each agency must maintain documentation showing the results of its review or audit and the plan for correcting material deficiencies revealed by the review or audit. To the extent that the audit documentation includes valuable formulate, designs, drawings, computer source codes, object codes or research data, or that disclosure of the audit documentation would be contrary to the public interest and would irreparably damage vital government functions, such audit documentation is exempt from public disclosure.

Credential Manager

Credentials are stored in the credential database. All actions concerning the credentials (including management of the credential database) are handled by the credential manager (CM). The CM checks the validity of the certificates, updates them, maintains the local revocation list etc. The local revocation list acts as a second black list only that this time the user can locally make invalid the agent's certificates and therefore force the system to treat the agent as an anonymous one. While the first list forbids migration to the agency (via SSL authentication) here we have only sandboxing of the agent (treated as possibly malicious). X509v3 Certificates [3] are used as credentials in a heterogeneous environment with a key used as the primary identification of a principal. In our approach we assume that users have certificates and that hosts also have certificates. Places can also have certificates in order to sign results. As the nested-place approach we take is service oriented (place n can belong to a different provider than the sub-place nix), we can ask from the nth place to sign a part of an agent. When checking the validity of certificates the credential manager looks up firstly his local database and his local revocation list. In the local databases a copy of the previous certificates of user's agents that have executed exist.

Component Manager

The Component manager mainly manages all requests concerning components preinstalled by the administrator as well as user installed components in the component database. The component manager allows first the administrator to install code and selectively via policy make it available to the users. This code can be signed so that agents coming to the agency can verify the originator of the code and decide whether to use it or not. This helps partially with the "Malicious Host" problem. Agents can decide if they trust the code they need in order to perform their goals. Furthermore the agents are able to verify a host before they migrate to it. So if every host n can verify host n+1 then we can make sure that our agent moves in a selected path of hosts. If the host is not trusted then the agent may decide not to execute there. User agents that are given permission can put their own code to this database and make it available to third party agents permanently or for a limited time. This increases the flexibility as well as the security and performance of the platform. The flexibility and performance because each user can have its own implementations of custom code on the node and thus his agents can be more lightweight and less complex. The component database can be considered a general database of active code, protocols, encryption algorithms, etc. Component database is of great significance to this approach as it ensures the up to date status of various components and also in parallel minimizes security risks for agents and for the platform. Security is by nature overhead in the communication and execution in order to protect the system.

Resource Manager

A resource manager is available in order to handle the resources assigned to the agency or place. We assume that resources are assigned from the administrator (that is the person that creates the place and this can be the agency administrator or one of the previous n-1 place administrators who created the nested place n) to a place n and are managed by the owner of the newly created place. The resources and their management are transparent to place users and to nested places that place n might contain. The place resource manager can handle the resources that are dedicated to a specific place. It can be contacted also directly via the agents that reside in the associated place also in the case that there is a need for more resources. Note that the resources available to a certain place are transparent to the agent and its users. This helps also with the Place Oriented Virtual Private Network (PO-VPN) [13]. In a PO-VPN scenario an enterprise can setup places spawned in a network infrastructure and therefore create a VPN of places where its agents can execute according to custom security policies and services.

Cache Manager

The cache (handled by the cache manager) is another essential part of the architecture and its usage is mainly focusing on improvement of the overall performance. Security checks are time and computing consuming processes. In our effort, not to duplicate all the time the necessary security checks, we have a cache. Security checks that have been done via the enforcement engine are stored with a time limit in the cache. If the time limit expires then the security checks are performed again, otherwise the security check is considered valid and is used by the system.

The policy DB can be dynamically updated via the enforcement engine any time. Thus the problem is faced that the cache contains outdated information. We solve this problem by deleting (each time the policy for an entity changes) the cached security checks that are associated with this key/person partially.

Audit Manager
Audit manager handles all audit events. Experience has shown that 100% security is difficult to realize - if not impossible - due to the multiple factors that interfere. Collecting data generated by network activity provides a useful tool in analyzing the existent security and also trace back (if possible) the originators of a security breakout. Having a detailed audit can lead to reconstruction of a sequence of events and better understanding of past security failures. Audit data include any attempt to achieve different security level or change entries in the system's databases etc. Intrusion attempts can also be detected via audit e.g. when we see repetitive failures in an attempt to use a component/service we can adapt our policy so that we prevent any possible intrusions.

7 The Need for Information Security Management Improvement

In a rapidly changing market place with pacing technological developments, pressure to reduce 'time to market' and demands for shorter 'security policies life-cycles' the result is a highly competitive information security environment. Security Policies who succeed in this environment are the ones that take their security schemas to deploying security more efficiently and effectively. Security Management is an activity of security deployment ownership from conception to standardisation. The Security Management function promotes efficiency, effectiveness and information assurance harmony throughout the organisation. In a security environment it is paramount for new security policies to achieve all expectations; successful Security Management is a key success factor in making this happens.

7.1 The Role of the Security Manager

Security Management is the security process that actively manages security policies s throughout their lifecycle. The Security Manager is the owner of the security schema in totality. Security success of the security schema is the ultimate goal of the Security Manager and to this end will require support from the organisation in order to succeed. Cross functional process management and an attitude of "making it happen" are key attributes for the Security Manager.

The Security Management activity should operate within the boundaries laid down in the strategic plan and as such is an integral part of the information technology function. The main objective of a Security Manager is to plan and specify a security policy/security portfolio in line with the long term strategic plan - Security Plans are a fundamental part of this process. Proposed security schemas must provide good synergy within the overall security portfolio i.e. security variants must be planned in accordance to meet security policies needs (Time to deployment) and within the scope and capabilities of the organisation.

8 Summary and Conclusions

Security architecture for agent based systems has been presented. This defensive model of design is focused on designing agent systems to be secure from the scratch. Adding security after the design phase has been shown to be difficult, expensive and inadequate. Security is not an explicitly called service and its treatment as such imposes further security risks in the infrastructure. We have showed that benefits such as simplicity, scalability, flexibility, interoperability, performance and safety have been addressed successfully. The components of the architecture have been analyzed and explained. Per identity/place security and customization as well as the rapid service creation is the main driving force for next generation mobile agent systems. In the future we intent to advance our approach. Our architecture tries to identify and prevent possible malicious agents. For the moment it can't handle collaborative attacks. Taking into account the tools provided (e.g. audit log, encryption tools, etc) one could implement stationary agents (guards) that reside on a place and based on intelligent internal strategy react to environment changes and try to track and eliminate collaborative attacks. Those guards could also work in collaboration thus providing a higher level of security to a number of hosts. As agent technology evolves and becomes more sophisticated a co-operative security infrastructure could be developed and deployed.

References

1. Cetus Links on Mobile Agents : http://www.cetus-links.org/oo_mobile_agents.html
2. F. Hohl, Protecting mobile agents with Blackbox security, Proc. 1997 WS Mobile Agents and security, Univ. of Maryland.
5. IETF Transport Layer Security (TLS) group http://www.consensus.com/ietf-tls/ietf-tls-home.html Protecting Mobile Agents against malicious hosts. Lecture Notes in Computer Science on Mobile Agent Security, November 1997. "Overview of Certification Systems: X.509, CA, PGP and SKIP", Meta-Certificate Group, Novware Softex/Unicamp Brazil.
9. B. Schneier and J. Kelsey, "Cryptographic Support for Secure Logs on Untrusted Machines", The 7th USENIX Security Symposium proceedings, USENIX Press, Jan 1998, pp. 53-62
11. OMG Web Site: http://www.omg.org/
12. FIPA Web Site: http://www.fipa.org/
13. Stamatis Karnouskos, Ingo Busse, Stefan Covaci, "Place-Oriented Virtual Private Networks", HICSS-33, January 4-7 2000, on the island of Maui, Hawaii.

14. Lightweight Directory Access Protocol (LDAP v3), RFC 2251. [15] Unified Modeling Language, Rational Software, URL : http://www.rational.com/uml
16. "Ten Risks of PKI: What You're Not Being Told About Public Key Infrastructure", C. Ellison and B. Schneier, Computer Security Journal, v 16, n 1, 2000, pp. 1-7.
17. Java Security Flaws http://kimera.cs.washington.edu/flaws/
18. MASIF - Mobile Agent System Interoperability Facility, http://www.omg.org/docs/orbos/98-03-09.pdf