Securing Mobile Agents for Survivable Systems

Jeffy Mwakalinga, Louise Yngström
Department of Computer and System Sciences,
Royal Institute of Technology/Stockholm University, Kista, Sweden
Fax: +46 8 703 9025 Tel: +46 8 16 1721
[jeffy@dsv.su.se, louise@dsv.su.se]

ABSTRACT
We have what we have today because of the decisions and actions that we made in the past. Our lives and computer technology in the future will depend on the decisions and actions we make today about them. In future, it is very likely that we will be walking with Web servers in mobile phones, PDAs, or MP3 players or in whatever devices. There will be so much information from banks, insurance, government, health, nursery, and schools requiring instant response that will necessitate people to carry Web servers. People will be required to make different authorization and privacy decisions, which cannot wait. The amount of information and actions can necessitate the need for helping hands in the form of mobile software agents, which are forms of non-human computer secretaries. These can be used in diverse business areas like auctions, contract negotiations, stock trading, and money transfer. These agents will need to carry information and perform transactions securely. How do we secure software mobile agents? In this paper, we describe ways of securing mobile agents for survivable systems. We describe ways of protecting mobile agents and the information that they carry.

Keywords
Software mobile agents, survivable systems, agent platforms, agent certifier, and accountability.

1 INTRODUCTION
The aim of this work is to study ways of securing software agents, which are used to perform different tasks during deterrence, protection, detection, response, and recovery services in the survivable systems. According to [14], “An agent is an encapsulated computer system situated in some environment and capable of reactive, pro-active, and autonomous action in that environment in order to meet its design objective.” An agent consists of three main components [3]: header, code, and a database. The header contains identity of the agent, agent attributes, signatures, travel paths, level of trust, ownership and other related information. The code section contains a system of programs performing the specific tasks of the agent. The database contains internal and the collected data while traversing in different environments. Agents are generated from an agent platform like Java Agent Development Framework (JADE) [15]. There are already software agents for different purposes. When one wants to find the best, ticket through the Internet to fly to a specified location it can take a lot of time and energy. To save time and
energy one can send a software mobile agent instead to do the job. Manufacturers of different products can negotiate prices, delivery of goods, terms of delivery and other services with supplies through their respective agents [4]. Other services suitable for mobile agents include network management, intrusion detection, testing security of networks and so on. The use of mobile agents reduces network traffic because they perform actions at agent servers reducing the request/reply messages in traditional client-server transactions. Mobile agents have to perform transactions and carry information securely. In this paper, we describe how to secure mobile agents for survivable systems. Survivable systems are those that are required to run all the time like air traffic systems, banking systems, medical systems, radars, and different business systems. To be able to run all the time they are required to have fault-tolerance measures. The methodology for building security in survivable systems is described in [13]. The Systemic-Holistic Approach [2] and the Immune system [1] paradigms are used as foundations in building security in survivable systems. The Systemic-holistic paradigm is used for studying security of a system as a whole by considering the system, the environment of the system and by considering technical and non-technical factors. The Immune system is used for protecting human bodies from different viruses and helps humans to survive in different environments. We study how living systems, particularly humans, survive in open environments and apply the features of the immune system to make systems survive. We use mobile agents in survivable systems and there are a number of security threats for mobile agents.

1.2 SECURITY THREATS FOR MOBILE AGENTS

Before addressing security, we need to understand the different security threats for mobile agents. The parties that are involved in transactions include agents and agent servers (platforms). An agent can attack an agent server, an agent server can attack an agent, an agent can attack another agent, an agent server can attack another agent server, and other outside attackers can cause security threats to the agents and agent servers [4].

Attacks from agents to agent servers include masquerading, denial of service and unauthorized access. Masquerading is a feature of an agent to pretend being another agent in order to gain unauthorized access of resources or to damage the reputation of the other agent and the owner of the agent. In denial of service, an agent disrupts the services offered by the agent server by running programs that heavily exploit system vulnerabilities of an agent server to degrade the performance of the agent server. Agent servers accommodate many mobile agents from different organizations. Some of the agents may try to access information on the agent server that they are not authorized to.

Attacks from agent to agent include masquerading, denial of service, repudiation and unauthorized access. An agent can exploit the weaknesses of another agent and steal its identity. The agent can then masquerade and perform any actions under other agent’s identity. Agents can launch denial of service attacks against each to prevent them intentionally from finishing their tasks.
An agent can cheat another agent to sign a bad contract and then repudiate later from having done that. An agent can change the information or programs in another agent if they are not secured. An agent can even call another agent’s methods in an attempt to change the behavior of the agent.

An agent server can attack a visiting agent in many forms: by masquerading, by denial of service, by reading agents information or by modifying agent’s information and programs. An agent can be cheated into paying higher prices for items that are being sold by an agent server. Outsiders can attack agent servers and agents by masquerading, unauthorized access, denial of service and by coping agents or parts of the agent messages and replaying them. After discussing security threats, we will discuss security requirements for mobile agents.

1.3 Security Requirements for Mobile Agents

According to [4] security requirements on agent frameworks include confidentiality, integrity, accountability, availability and anonymity. Confidentiality is required so that all the classified information can be kept secret at agent platforms and while being carried by the agents. Communications between agents and between agents and agent servers should also be confidential. All messages’ flow should be kept secret so that the listeners should not be able to find out the number of messages nor analyze the traffic between agents and platforms. Even the location of agents should be confidential. Agents can choose to be public and in such cases, they should be allowed to be. The activities of agents should also remain confidential so the audit logs of their activities must be protected.

Integrity of agents’ code, state, internal data and collected data should be provided to ensure that unauthorized modification of code, state and data is not done. Agents should be able to detect when modification of their code, state and data is done. The agent server must also be provided with integrity. Access control should also be addressed so that only authorized agents should be able to access and perform the tasks on agent servers. Changes to agent servers should be made only by authorized users.

Accountability, according to [4], includes identification, authentication and audit of human users, agents and agent servers. This includes maintaining records of security related events of user/agent name, access to objects, time of access, type of event, success or failure of event. Audit logs will force users and agents to be accountable for their actions making it difficult for them to deny having performed the actions. Audit trails of agents should also be kept to help tracing activities in case of errors. Agents and agent server must authenticate each other before performing any transactions. Authentication could be strong or simple, depending on the classification of transactions. When agents are accessing public information, agent servers may not require any verification of identities of agents.

Availability of information and services to mobile agents must be ensured. The agent servers must support simultaneous access, allocate resources fairly, be able to recover from different failures and so they should have fault-tolerance measures. Agent servers should scale and be able to handle requests from many agents. When the agent servers are
not able to provide this service, they should notify agents about it. Denial of services attacks from malicious agents or other sources on the agent servers should be strongly authenticated before performing them. The agent server should have a balance of the need for the agent to be anonymous and the need for the platform to hold the mobile agent accountable for its actions. The agent server can keep the identity of an agent and its actions secret from other agents as long as the agent is behaving in accordance to the policies and security requirements of the agent server but when the agent crosses the red line it will be revealed to other agents.

1.4 ORGANIZATION OF SECTIONS

Section 2 covers related work. Section 3 describes the security architecture of survivable systems. Section 4 discusses agent security. Section 5 briefly discusses conclusions.

2 RELATED WORKS

In [3] a comprehensive security infrastructure for mobile agents is described. The infrastructure provides authentication, authorization, integrity, accountability and non-repudiation. Authenticity of agents is provided by giving identities to agents. The Agent identity has static, dynamic identity and other specific identities. Static identity comprises of agent author’s ID (author’s certificate), agent owners ID (owners certificate) and agents name. Dynamic identity consists of agent home ID and time of launch. To verify identities one verifies the certificates. Authorization of agents is provided through agent attributes, which contain level of trust, agent task specifications, constraints of agents, agent owner credentials. Constraints on agents include expire-time, maximum size, whether an agent can create children, and others extensions. Integrity is a security service for making sure that information is not modified when on storage or on transmission. Integrity of agents and agent servers is provided through digital signatures. The signatures that every agent must have include agent authors, agent owners, trusted appraisal’s, privilege authority’s, sender’s, agent server’s signatures. Confidential information that is carried by agents is kept secret from other agents.

FIGURE 39: MOBILE AGENT COMPUTING MODEL

The lifecycle of an agent includes creation, owning, launching, traversing, hosting and returning home as shown in Figure 39. The agent developer creates an agent signs it and attaches the digital certificate. The agent is then sent to the trust appraisal that verifies the signatures, tests the agent and then puts a level of trust on the agent. She signs the agent and puts her certificate. The agent is then sent to the owner who had requested it. The owner verifies the
signatures of the agent developer and the trust appraisal. If successful, she accepts the agent. The owner assigns agent identity to distinguish it from other agents.

Before launching the agent, the owner writes specifications on the agent, gives constraints to the agent of lifetime, maximum size and other specified properties. The owner then assigns the home address, destination server, time of launch. She then signs the agent, seals it with the destination server’s public key, and sends the agent.

The destination server opens the seal, verifies the signatures of the author, trust appraisal and of the owner. If verification is successful, she accepts the agent. The server-hosting agent protects its information that is classified from the visiting agent. The server monitors the actions of mobile agent. Information collected from the agent server is sealed by the owner’s public key and then it is signed. The hash of the state of the agent is sent to the state server. The agent can then be sent home or to another agent server and the procedure before sending an agent is the same as when the owner was sending the agent to the destination server. When the agent arrives home to its owner, the signatures are verified the state of the agent is checked. If something has gone wrong, the owner extracts hashes of states from the state server and traces the whole communication. This system provides most of security services in accordance to the security requirements, which were discussed in sections 1.3, confidentiality, integrity and accountability. The limitation of this system is that it does not provide anonymity and availability requirements.

3 SECURITY ARCHITECTURE FOR SURVIVABLE SYSTEMS

In [13] we developed a methodology for security survivable systems and the architecture for these systems is shown is Figure 40. The components in the architecture of survivable system include the deterrence, protection, detection, response and recovery sub-systems. It also includes an administration component containing the agent generation library, a system manager, a database, an integrated security system, special analysis component and the system's fault tolerance manager. The fault tolerance manager detects errors, assesses the damage, and confines the damage, performs error recovery measures, does fault treatment measures, locates the errors and performs measures for continued service. Every sub-system has sections: inputs, process, outputs, and fault tolerance manager. The sub-systems also have memory and feedback mechanisms for analyzing and modifying inputs when necessary.

3.1 DETERRENCE SUB-SYSTEM

The deterrence sub-system is aimed at scaring off attackers (like how a cat scares off attackers by increasing its size and through fierce screams). When criminals plan to rob a bank in the physical world they do surveillance of the bank to determine whether it is possible to attack, take what they want and get out without being caught and without living evidence. In the digital world, the attackers do more or less the same. Before would be attackers intrude a system, they do some kind of scanning to determine the operating systems and their versions, the ports that are open, the applications and versions that and on the victim’s system.
Then the attackers do possibly also social engineering to understand the architecture of the system. There are many ways of doing this, from just asking the people working there to listening to conversations of system administrators there or secretaries working there. It is surprising how employees like to talk about their jobs during lunches and even dinners! From the results of scanning and social engineering, the criminals decide whether it is possible to attack the system, and get out without being caught and without living evidence. The attackers will not attack a system if it is considered too risky. The functions of the deterrence sub-system include: adapting to the new and unknown surveillance methods; organizing training to prevent social engineering; monitoring surveillance attempts; redirecting attacks to specialized environments (like honey pot system); handling replies to scanners (returning nothing, a warning, etc); auditing; tracing scanning sources.

3.2 PROTECTION SUB-SYSTEM

Protection sub-system has measures for guiding the territory of a system and its entities. Home cats establish territories, a special place on a sofa, and put rules. Wild cats mark territories by using peculiar identifying items like natural scents. The protection sub-system provides security services: authentication, integrity, confidentiality, non-repudiation and authorization of entities and information during storage, transmission, processing, collection and display. Other features of this sub-system include adaptability in which the system learns new protection ways by applying the latest standards; organizational, like configurations in accordance to the security policy; semi-autonomy in which the system makes some decisions without involving the management of the system, but the critical decisions must involve the system management. Multi-layer protection is a feature where protection is provided at the boundary of a system and inside the system and sub-systems. Another feature is partial distribution – in which protection is done locally while in some cases protection is coordinated.

3.3 DETECTION SUB-SYSTEM

This sub-system is responsible for detecting the abnormalities, storing and protecting the log of events, analyzing the events, monitoring, managing and interacting with other subsystems. Other features include multiple-layer detection, adaptability of new ways of monitoring and detecting, semi-autonomous, and dynamic coverage, sending reports to the database and the administration. The normal behaviors of outgoing and incoming messages are defined. Software agents are used to detect the abnormal behaviors of incoming and outgoing messages, as cells are used to detect foreign cells in immune systems. All the entities that belong to a system are labeled as ‘self’ by being given special identities and being registered in a database. Software agents monitor a system to discover the non-self entities in a system.

3.4 RESPONSE SUB-SYSTEM

This sub-system is responsible for incident management. It classifies incidents into false alarms, minor and major incidents in accordance with the security policy of the system. The response and speed of reaction depends on the classification. It makes decisions on how to respond for every incident. The decisions include disconnecting the affected sub-system from others,
slowing, shutting down or restarting the affected system, etc. The sub-system also sends reports to the affected users, to the database and to the administration. Other functions of this sub-system include managing patches and adaptability, tracing the attack, mitigation of the attack and so on.

3.5 Recovery Sub-system

The recovery sub-system is for bringing an attacked system back to normal. The functions of this sub-system include managing back-ups, re-installing the programs, periodic and emergency vulnerability testing, restoring a system from back-ups, collecting and protecting evidence, fixing the vulnerabilities. The agents can help to define and test business continuity plans. This process can be very expensive and takes much time if done manually. At every moment, three types of the state of system and sub-systems and operations are stored: the original state, the intended state, and the actual state. When an incident occurs, the system can go back to the original state and flush all the rest. This feature could be partially or wholly implemented. This depends on the current technology and other back-up resources.

3.5 Other Components

The integrated security system is used for certificate management, managing authorization and provides smart cards, database and information protection services. The special analysis component is used for analyzing inputs and other objects that are not understood by the sub-systems. The system fault-tolerance manager is responsible for the overall fault-tolerance of the whole system. It also controls the fault tolerance managers of the sub-systems. The system manager is responsible for managing all the operations of the system. This includes configurations, communications with other systems, controlling the all the components. All these sub-systems have fault tolerance managers which have error detection measures; damage assessment measures; damage confinement measures; error recovery measures; fault treatment and locator and continued service measures.
FIGURE 40: GENERIC MODEL FOR SURVIVABLE SYSTEMS
FIGURE 41: AGENTS

Agent
Author

Agent
Certifier

Agent
Owner

Hosting
Agent
Server

Hosting
Agent
Server

FIGURE 42: AGENT STATIONS
4 AGENTS SECURITY

4.1 OVERVIEW

The lifecycle of software mobile agent starts at an agent author, then it goes to an agent certifier, thereafter it goes to an owner, and then it is launched to different servers to perform the tasks specified [3] as shown in Figure 42. The agent developer and owner could be the same but the agent certifier and author/owner are not the same. An owner sends a request to an agent developer with task specifications of the agent. The agent developer creates the agent, comprising of a header, code and data [3]. The header contains the identity, attributes, recipient information, travel path and signatures. The data section is divided into internal data and collected data. The identity of the agent has three main parts [3]: static identity; dynamic identity, and other specific identities. The static identity contains the author’s ID, owner’s ID and the ID of the agent. The dynamic identity contains the agent home and time of launch. Other specific identities can include digital certificates and other tokens. Attributes of the agent include level of trust, task specifications of the agent, constraints of the agent, and credentials of the agent owner.

Signatures include agent authors, certifier’s, privilege authority’s (this authority issues agent’s security credentials to users), agent owners, agent sender’s signatures, and agent server’s signatures. Collected information include is information from different servers where an agent has been visiting. Agents are created by an agent developer according the task specifications. The tasks are specified by the one requesting the services who then signs the agent

4.2 PROTECTING THE AGENT SERVER

There are a number of technologies [4] for protecting agent servers. One of this is called Software-based Fault Isolation (sandboxing) [5]. This is when untrusted agents are isolated and monitored in a special environment. When other agents, which are not part of the protection system, come to an agent server they will be authenticated and put in different domains or sandboxed depending on the trust level of the agents. The second technology is known as Safe code Interpretation [4] that means that a command that is harmful can be made safe or denied execution. Many agents today are created in using interpretative programming languages, like java, that are platform independent and scripts to be able to run in all platforms. Another technology is called signed code in which agents and other objects are signed digitally by private keys. A digital signature enables the agent server to verify the identity of an agent, the origin of the agent and its integrity. Java applets can be signed, which enables them to perform actions in a wider range of platforms.

Another technique is called State Appraisal [6], which is a way of verifying the correct state of an agent before accepting the agent and before authorizing the agent to access objects. Path Histories [7] is another technology, which aims at making sure that the agent servers that were visited before the current platform are authentic and have agent servers sign the information collected by the agent. Another technology is known as proof carrying code [8], which is a way of forcing authors of agents to prove that they have included safe measures in designing and creating agents. The proof and the code are sent together to the consumer where it can be verified in a simple way without using complicated cryptographic measures and without needing any help.

In this work, we apply Signed Code, Path Histories, a form of State Appraisal and a form of Sandboxing. The agents are signed by both the creator of the agent, the verifier, the owner and the sender of the agent. In this way, we can verify the identity of the agent, the home platform, the sender and the verifier of the code. Path Histories’ method is used by having
the servers, where the agent is visiting, sign the information collected. State appraisal is done not by the agent server but by the certifier of the agent where a trust level is specified so that the hosting agent can decide in which category to put the agent. Sandboxing is applied to agents that are not from the protecting system. Next, we discuss how agents are protected.

4.3 Protecting the Agent
Protecting agents is different from protecting agent servers [4] because the agents do not have their own processors and they cannot extend the home platform, but have to rely on the environments provided for them. Protecting agents is more of a detective and deterrent manner while protecting agent servers is preventive, detective, and deterrent. There exist a number of technologies for protecting agents [4]. One of them is called Partial Result Encapsulation, in which the results from each visited agent server are encapsulated. This can be done by the agent or by the agent server. However, it is recommended to be performed by the agent itself. One way that can be applied is called sliding encryption [9] in which the agent seals information every time it collects it. The agent can use the public key of the owner to seal the information, so that only when the agent returns home that the collected information is unsealed.

Another technology is known as Mutual Itinerary Recording [10] in which two cooperating agents record and track each other’s movements by sending to each through a secure channel the last agent server, the current and the next agent server. The next technique is called Itinerary Recording with Replication and Voting [11], which is similar to Path Histories [7], but it has been extended with fault-tolerant measures. There are multiple copies of an agent doing the same tasks. This method is resource demanding. The next technology is called Environmental Key Generation [12]. This is a way of protection in which an agent generates a key and protects all the executables if some environmental conditions are true. In this work, we use Partial Result Encapsulation as described in section 4.7. Details of security services in different scenarios are described in the following sections.

The mobile agents that are performing fault-tolerance tasks have special security properties. They are authorized to access agents and inspect the agent headers, agent codes, and data to detect errors, assess damages and so on.

4.4 Security Services During Agent Creation
For survivable systems, the agent generation library, shown Figure 41, is the agent author. In future, agents could be purchased / requested from other qualified authors. The sub-systems deterrence, protection, detection, response and recovery are the agent owners. These sub-systems request agents from the agent generation library in accordance to their specifications as shown in Figure 41. The special analyzer acts as an agent certifier, but in future there could be an independent body for certification of agents. The sub-system, for instance Deterrence, verifies the agent generation library before requesting a mobile agent. After successful authentication, the denial of service cookies will be shared between the sender and the destination. These are functions of an address and a secret key. These will be part of all the communications between these parties. The aim of denial of service cookies is to reduce denial of service attacks. Communications that do not have denial of service cookies attached with specified properties are ignored. These cookies are not like the normal cookies that servers give to client browsers when visiting their sites.

The agent generation library verifies the identity of the particular sub-system. If the verification is successful, the sub-system requests the required agent for its tasks from
the agent generation library. Every sub-system has many different agents for doing diverse kinds of tasks in this sub-system. The agent generation library composes the code. The agent generation library calculates the integrity of the code and then the separate integrities of the header and the data and attaches its digital certificates. Note that the private keys of the agent’s author, certifier and owner are never stored in the agent. To provide authenticity of agents the agent generation library signs the agent. To provide confidentiality requirements, the agent creator seals the agent by using the public key of the special analyzer, which is acting as the agent certifier. The agent is then sent to the special analyzer. The special analyzer opens the message by the private key and verifies the signatures and the integrity of the agent. If successful the special analyzer checks whether the agent is behaving in accordance to the specifications. The certifier puts a trust level and its digital certificate. The analyzer signs the agent, protects it and sends it to the sub-system. The sub-system opens the message using its private key. It verifies the signatures of the agent creator and of the special analyzer. The sub-system then notifies the agent creator and the certifier that it has received the agent. The sub-system also puts authorization attributes like mobility, expiration time, size limit of data it can collect and whether the agent can create (spawn) children. The sub-system’s controller acts as a privilege authority and issues credentials like roles, group membership and monitoring attributes.

4.5 SECURITY SERVICES DURING AGENT LAUNCHING

The agent can be operating locally or it can be sent to deter, detect, protect at other locations of the system. Before being sent to the location, the sender does the following procedure. The sender specifies the tasks of the agent; Assigns the dynamic identity by adding agent home ID and time of launch for authentication purposes. The sender also attaches the digital certificate of the sender; the owner’s signature of the agent is added for providing integrity; Sends the state of the agent to the controller of the sub-system for audit trails.

The signature of the sender is calculated by putting the receiver’s address, adding the hash of the agents’ state, adding the timestamp and a random number. All this information is put in the recipient information field and is then signed. To provide confidentiality the whole message is sealed by the destination server’s public key. The sender and the receiver authenticate each other before sending the agent. After successful authentication, the denial of service cookies will be shared between the sender and the destination.

4.6 SECURITY SERVICES DURING AGENT HOSTING

According to [4] the agent server should provide separate domains for each agent that it is hosting, but in this work we don’t provide separate domains for the security agents because they are deterring, detecting and protecting the system and they are supposed to move freely. When the destination server receives the agent, it does the following procedure:

Opens the agent using its private key.
Verifies the agent generation library’s and certifier’s signatures to check for integrity.
Verifies agent owner’s digital certificate
Verifies agent owner’s signature
Verifies agent sender’s signature
Checks the time stamp, hash of state and intended recipient in the recipient’s information. If successful the server accepts the agent, monitors the agent to provide accountability requirement and the audit logs are protected by the agent server.

When the agent has done the tasks the agent server will sign the information and send the agent home or to the next agent server as described in section 4.7.
The agents that are not from the security system will be sandboxed if they are not fully trusted. This will reduce the denial of service threat from these agents. In cases where denial of service is launched by outside attackers, the address of the agent platform will be temporarily changed until the problem has been solved. The agents that require anonymity will have their identities hidden from other agents.

4.7 SENDING AN AGENT FOR CLONING

If the agent is very successful in deterring, protecting, detecting intrusions and other tasks in accordance to the specified criteria the agent will be sent to the agent generation library for cloning. In sending the agent, the following procedure will be followed:

The agent generation library and the sender will authenticate each other before sending the agent.

Attach owner’s digital certificate.

Assign agent home ID and a time stamp which is the dynamic identity for authenticity purposes.

Create the owner’s signature for integrity requirement.

Create sender’s signature by putting the receiver’s address, adding the hash of the agents’ state, adding the timestamp and a random number. All this information is put in the recipient information field and is then signed.

The whole message is sealed by the agent generation library’s public key to provide confidentiality and then it is sent.

When the agent generation library receives the agent it will perform the procedure in section 4.6 and will then clone the agent and will send the agent back through the agent certifier as described in section 4.4. A copy of the agent is stored in the database of the agent generation library.

5 CONCLUSIONS

In this work, we have provided ways of securing agents for survivable systems. The security requirements confidentiality, integrity, accountability are met. Information carried by agents and that, which is stored at agent servers, is kept confidential. Communications between agents and agent owners and agent servers are protected. Integrity of agents and data is provided through signatures. Accountability is provided through monitoring, signatures and log files protection.

Denial of service is partially addressed by using denial of service cookies and by sandboxing, untrusted agents. Anonymity is not complete; a mobile agent can be anonym to other agents but not to the agent server. An agent server has the right to monitor an agent. Limitation is that availability and anonymity requirements are partially met. Future work will be to implement the agent security. Agents that are used for fault-tolerant have all the authority to access agents for inspection purposes.

REFERENCES


