Trust Enhanced Security - A New Philosophy for Secure Collaboration of Mobile Agents

Ching Lin and Vijay Varadharajan
Department of Computing Macquarie University
North Ryde, NSW 2109, Australia
{linc,vijay}@ics.mq.edu.au

Abstract— **The mobile agent computing model violates some of the fundamental assumptions of conventional security techniques. Consequently, this has rendered many of the existing conventional security countermeasures less effective for mobile agents. In this paper, we propose a new philosophy of trust enhanced security, which advocates a paradigm shift for mobile agent security solutions: from security-centric to trust-centric with the aim of providing improved security and performance of mobile agents. We first examine the problem of uncertainty in behavior induced by the security assumption violations by mobile agents; we then propose a trust enhanced security approach and argue for the need for a paradigm shift to trust-centric solutions. Next we identify a list of general design requirements for the trust-centric solutions and outline the new architectural design which supports the new trust enhanced security philosophy in practice. Finally we discuss the emergent properties of the new architecture and introduce the experimental results for validating the properties.**

I. INTRODUCTION

The mobile agent model is a more recent paradigm for distributed computing compared to the traditional client-server model. It offers an opportunity to achieve the vision of usable distributed systems in a heterogeneous network environment such as the Internet. Mobile agents are autonomous programs which migrate through a network of agent enabled hosts to accomplish tasks on behalf of their owners. An agent can suspend its execution on a computer at one host, transfer itself to a new host and resume execution on the new host. This ability to move computation across the nodes on a wide area network helps to achieve the deployment of services and applications in a more flexible, dynamic and customizable way than the traditional client-server model [1].

However, while the mobile agent paradigm offers these attractive features to distributed applications, it gives rise to a range of new security challenges [2], [3]. Just as in any distributed system, when a request for a certain service is received by one principal from another, the receiving principal needs to address at least two questions. Is the requesting principal the one it claims to be and does the requesting principal have appropriate privileges for the requested service? These two questions relate to the issues of authentication and authorization. There are also other security concerns such as auditing, secure communication, availability and accountability. When it comes to mobile agents, the security issues become further complicated. First, there is a greater opportunity for abuse and misuse. Second, mobile agents introduce specific issues that are unique, which challenge some of the common assumptions that are often made in secure systems design. For instance, it is not always easy to identify a particular mobile agent with a particular known principal. This in turn makes it more difficult to associate the privileges with the mobile agent and enforce the security policy. In fact, the development of comprehensive security solutions is a major concern that needs to be addressed before we can see a wider industry adoption of the mobile agent computing model for many network based applications [4], [5]. In this paper, we propose a new philosophy of trust enhanced security for developing a new class of security solutions for mobile agents. This new philosophy outlines guiding principles for integrating trust into security and provides a framework for making better security solutions in the mobile agent context. In particular, it aims to improve the security performance of the primitive security operations of mobile agents such as itinerary composition and authorization. We outline the design requirements of a new architecture to support this philosophy in practice. Furthermore we discuss emergent properties of the new architecture and introduce the experimental validation results for these properties.

Section II introduces the security issues of mobile agent. Section III argues for the relevant of trust in mobile agent security. Section IV discusses the paradigm shift to trust enhanced security. Section V outlines the new architectural design. Section VI discusses the emergent properties of the new architecture and introduces the experimental results. Section VII concludes the paper.

II. SECURITY ISSUES OF MOBILE AGENTS

A. Security Threats

There is a variety of ways of classifying security threats in the mobile agent environment. We consider these in terms of agents attacking the agent host, agent host attacking the agents, agents attacking each other in a host and attacks against the agents when they are transferred over the network. In each of these categories, a variety of common security threats such as masquerading, unauthorise access, unauthorise disclosure, unauthorise modification and non-repudiation can arise.

B. Countermeasures

Many conventional security techniques used in contemporary distributed applications (e.g. client-server) are in general applicable within the mobile agent context, though certain

1-4244-0429-0/06/$20.00 ©2006 IEEE

special precautions need to be taken due to the unique characteristics of mobile agents.

Protecting an Agent Host. Without adequate protection, an agent host is vulnerable to attacks from many sources including malicious agents, malicious hosts, and other malicious entities. Fortunately most conventional protection techniques can be used to provide protection for the agent host. That is, within the mobile agent paradigm, the agent host is a counterpart to a trusted host within the traditional framework. Conventional security techniques include the following: 1) mechanisms to isolate processes from one another and from the control process, 2) mechanisms to control access to computational resources, 3) cryptographic methods to encrypt information exchanges, 4) cryptographic methods to identify and authenticate users, agents and hosts, and 5) mechanisms to audit security relevant events occurring at the agent host.

Protecting an Agent. Because an agent is a software entity, the traditional view that hardware protects software applies only when the set of hosts an agent visits can be trusted to some degree. Assuming an agent trusts its home base host (i.e. an owner host) to provide the required support services and not subvert its activities, countermeasures in the form of conventional security techniques can be applied on behalf of the agent via the host. These measures rely primarily on identifying and authenticating trusted parties with whom to inter-operate and include the following: 1) issue users and agent hosts public key certificates for strong authentication, 2) communicate information (e.g. agents and messages) securely among agent hosts (i.e. with confidentiality, integrity, source authentication, and non-repudiation), 3) detect and prevent replay attacks against agent hosts, and 4) enable an agent to audit host services and other security related events for post processing analysis and detection. A detailed review on mobile agent security issues can be found in [6], [7].

C. Drawbacks of the Current Solutions

There are three factors that influence the effectiveness of the security solutions for mobile agents. These are the mobile agent operating environment, violation of security assumptions and the limitation of "security-centric" centric solutions.

The Operating Environment. The operating environment of the mobile agent systems often takes two forms: 1) as a traditional closed system which has well defined security hierarchies (such as an Intranet within a business organization), or 2) as an open network environment such as the Internet, where no single fixed form of security hierarchy exists.

Violation of Security Assumptions. The mobile agent computing model violates some of the fundamental assumptions of conventional security techniques [2]. The most important assumptions are the identity and intention assumptions. The identity assumption states that "whenever a program attempts some action, we can easily identify a person to whom that action can be attributed, and it is safe to assume that that person intends the action to be taken". Given the identity assumption, the assumption about intention can be stated as follows: "we can say something about the sources of most security threats - significant security threats come from attackers running programs with the intent of accomplishing unauthorise results" [2]. For instance, the scenarios involving attackers masquerading as someone else in order to take actions that they themselves are not permitted to take, or legitimate users exploiting bugs in the security system to allow their programs to do things that violate the system's security policy.

In an open network operation, without a carefully-designed infrastructure, it may be impossible to identify the author or sender of the program; even if an individual or organization can be positively identified, there may be no way to judge how much trust to assign to them. The identity assumption is therefore violated. Consequently when a program attempts some action, we may be unable to identify a person to whom that action can be attributed and it is not safe to assume that any particular person intends the action to be taken. In an implementation sense, this translates into the following: when a program attempts some action, we cannot determine whether or not the action should be permitted by simply consulting the rights that have been granted to the user running the program, since the program may well not reflect the intent of that user. This considerably complicates the issue of security in mobile agent systems, since we can no longer do the obvious authorization check to see if an action should be permitted. Even when we can identify a principal to whom the program's actions should be attributed, that principal may not be known; that is, the principal relevant to determining the trustworthiness of a program may be someone not known to the system. Chess [2] suggests that we must either have a way of finding out things about users who are unknown to the system, or lump all unknown users together into one (presumably completely untrusted) class. The first suggestion is difficult to do as security systems are not geared up for this task. The second suggestion puts up an artificially high barrier to interacting entities which will reduce the system utility.

A natural consequence of the violation of identity assumption is that the intention assumption will be violated. This is because, under the traditional intention assumption, once we have satisfied ourselves who the principal (user) is (i.e. the identity assumption), we can then make sure that programs run by that principal (user) can do only those things that that user is allowed to do through the standard authorization or access control process.

Limitations of security-centric solutions. In general the currently existing security solutions can referred to as "security-centric", based on the conventional security techniques relying on cryptography based mechanisms to provide security services to the mobile agents. Security-centric solutions work well for the traditional closed environment where reasonable confidence can be gained about an entity's behavior once the entity is properly authenticated. Hence there is a perceived link between the identified entity and its expected behavior under this condition. However, its performance in an open network environment is much reduced. This is because of the uncertainty in the identities and their intentions in an open

network and there is no guarantee that a trusted third party (TTP) will be available to vouch on behalf of the interacting entities. Without certainty in either the identity or intention, no direct link can be inferred with confidence between an entity and the expected behavior. Security-centric solutions are hence rendered less effective when used in an open network environment. for instance, just because an agent carries a signed public-key certificate, it does not mean that the owner of such an agent is not an industrial spy with malicious intent. Nor does it necessarily mean that the receiving host can be sure that the requesting agent will not behave maliciously (e.g. the mobile agent code does not contain viruses or Trojan horses).

D. Problem Statement

A comprehensive mobile agent security solution should have the ability to handle both crypto-based and behavior based evidence. Such a solution should allow for the lack of certainty induced by the violation of the identity and intention assumptions. Just as in a human society, mobile agent entities can trust each other to different degrees which can be evaluated to allow appropriate interactions. Such a process can be facilitated by complementing the crypto-based evidence with the behavior based evidence. On the other hand, without the support of crypto-based evidence, traceability of the system will be poor, and the desirable conventional security properties such as integrity, confidentiality and non-repudiation cannot be maintained.

Let us now briefly highlight the above issues in some practical scenarios of counteracting the attacks introduced in the previous section:

In terms of an attack against a host (e.g. in the form of masquerading), the threat is one of "an agent claiming to be a different identity". Often in security, when a program attempts a certain action, its identity is related to the principal (e.g. a user) who is requesting that action to be performed. In the case of mobile agents, this may be the principal who is sending the agent. Knowing that an agent comes from a particular sender is often not adequate when it comes to determining the level of *trust* that can be placed by the host on the agent. What may be required is that the principal that one *trusts* is the one who has created that particular agent. So it may be required to authenticate both the sender and the creator principals of the agent. Furthermore, often in mobile agent systems, many programs are obtained from unknown or *untrusted* sources. For instance, when an executing host receives a visiting agent, it may not be clear as to what *level of trust* can be placed on the source from which the program comes. On the other hand, the sender of the mobile agent may also wish to authenticate the visiting agent host where the agent is to be executed prior to sending the agent. Hence mutual authentication is required in peer-to-peer transactions.

Having authenticated the mobile agent, the agent host needs to determine what actions the mobile agent is allowed to perform and whether it has the necessary privileges to carry them out. In general, the authorization decision for a mobile agent to perform a certain action can be based on a combination of privileges such as the privileges of the creator of the mobile agent and of the sender of the mobile agent as well as the function of the program code and the state of the agent. The authorization mechanisms control the behavior of the agent within the agent host thereby allowing protection of local resources. However as we mentioned above given that it may be difficult to identify the principal to whom the action can be attributed, it poses difficulties in determining whether or not the action should be permitted.

An even more difficult issue is that of protecting the agent from a malicious agent host. This is because the agent host provides and controls the computational environment in which the agent operates. A malicious agent host can modify the agents code and state and hence can affect the running of the agent. It can introduce unacceptable delays or simply not execute the code or even terminate the agent. Furthermore, we view agent itinerary composition as an important process in agent protection. Once a malicious host is included in a itinerary, no amount of cryptography will be able to protect the agent. This can be costly for both the agent owner and other benevolent hosts in the itinerary. On the other hand, when an itinerary is composed using *trust* information based on past behavior (e.g. behavior based evidence), it has the opportunity to remove certain malicious hosts at the composition stage, thereby offering the potential to improve the agent security performance *a priori*. Unfortunately most of the existing mobile agent systems lack such measures.

We claim that the above problems are largely due to the lack of ability in managing security decisions with reference to dynamic behaviors of the mobile agent system entities. As the concept of *"trustworthiness"* is increasingly used for quantifying the dynamic behavior in general security system designs, we believe that these problems can be mitigated by approaching them from a *"trust perspective"*, where the dynamic behavior of the system entities can be captured and managed through trust and through the integration of trust into the security decision making process. Naturally, such an approach needs a new security architecture to be developed to accommodate the interaction between security and trust. Ideally, such a new security architecture needs to include a trust management layer, in addition to the security management layer (characterized by the existing security solutions). This enables the management of the relevant trust relationships and provides the ability of trust enhanced security decisions. The hypothesis is that by factoring in the trust based behavioral evidence to the security decision making process, the accuracy of the security decisions will improve. This in turn leads to the development of what we refer to as a "trust enhanced security" approach.

III. Trust and Security in Mobile Agent Context

Intuitively, we assume that trust in an entity can be thought of as a combination of honesty, competence, reliability and availability of the entity with respect to interaction and co-operation with other system entities. In the context of mobile agents, we refine this broad trust concept into three important

classes of trust, namely, *Authentication Trust*, *Execution Trust* and *Code Trust*, which we believe to be relevant for the host and agent security issues. Since our discussion in this section will be of a high level nature, we only use the intuitive meanings of these trust concepts for the benefit of clarity (formal treatments of the trust concept in mobile agents are provided fully in [8].)

In reviewing the existing trust models [8], we have noticed that while researchers agree that trust is an important notion in security issues, there exists a gap between this consensus and the development of an approach that utilizes trust for security performance enhancement. This paper aims at closing this gap by proposing a new philosophy of promoting the integration of trust into security for better security decision making and hence security performance improvement. Next let us take a closer look at uncertainty induced security issues and initiate our effort towards using trust decisions to counteract these issues.

The uncertainty of identity. We believe that this uncertainty can be mitigated by improving the authentication process using trust in the authenticity of an identity (i.e. *Authentication Trust*). Here, we consider that each entity is asking the question about how much it trusts other entities in terms of authenticity, before proceeding with any other primitive operations - such as mobile agent itinerary composition and authorization. Due to the uncertainty of identity in mobile agent context, we can no longer rely on the conventional authentication process to give us a definite answer. Hence we propose to use a relevant trust relationship (i.e. the *Authentication Trust*) in the evaluation of the authenticity of an identity in an attempt to reduce the uncertainty.

The uncertainty about intentions. Again we believe that mitigation of this uncertainty is possible by using the *Execution Trust* and *Code Trust* to improve both the itinerary composition and the authorization process.

From the mobile agent owner's point of view, we need to have trust in execution (i.e. *Execution Trust*) in order to compose a secure itinerary for mobile agents. Execution trust is the trust the owner has in other execution hosts that they will faithfully run the mobile code without tampering and then migrate it to the next destination. Only through the evaluation of such trust, can an agent owner host gain certain confidence in composing a particular execution host into the itinerary of the agent.

From the executing host point of view, we need to have trust in the mobile code (*Code Trust*) to improve the authorization process; this is impacted upon by the intention uncertainty. Code trust is based on the ability of the agent owner host (or agent creator principal) to produce a good code and on the honesty of the prior executing hosts on the itinerary who have executed the agent for not tampering with it or making it malicious. Through the evaluation of such trust, the execution host can gain certainty confidence while receiving an agent.

IV. Paradigm Shift to Trust Centric Solutions

In order to make better security decisions in the face of uncertainty, we need to address the above security related trust issues. We believe that through the evaluation of these relevant trust relationships and subsequently trust decisions, we can improve security decision accuracy. The reason is that we can now take into account the dynamic behaviorial aspects of the underlying mobile agent entities. To this end, we propose a new philosophy of trust enhanced security which advocates a paradigm shift from the "Security-Centric" to the so-called "Trust-Centric" solutions. Derived from the Greek word "paradeigma", meaning model, a paradigm is a way of looking at and understanding the world. In our context, the new paradigm means a pattern or a set of models that influences the way mobile agent security decisions are made. In our context, this means the integration of trust into security through forming trust decisions as part of security. We believe that this paradigm shift from security-centric to trust-centric solutions is a important step in the search for more effective security solutions for mobile agents.

A. Trust-Centric Solutions

Why trust-centric solutions? The main reason for developing a trust-centric solution is that security-centric solutions cannot guarantee behavior of system entities (cf. Section II). In a security-centric solution, a malicious entity could seem perfectly normal even with all the crypto-based evidence in place, due to the lack of linkage between crypto-based evidence and the actual behavior of the entity in an open network environment. In general, the root cause can be traced to a lack of ability to capture the aspects of "trustworthiness" on the behavior of the entities in the underlying mobile agent systems.

What are trust-centric solutions? We refer to trust-centric solutions as a new class of security solutions using behavioral evidence in conjunction with crypto-based evidence for making improved security decisions. While crypto-based evidence can be generated from security-centric solutions, behavioral evidence (in the form of security related trust relationships) is captured by trust models. These trust relationships are locally managed and globally disseminated to facilitate the distributed security decision making process of mobile agent systems. The trust decisions based on the evaluation of this evidence have the ability to reflect the security related behaviors of the associated entities. Forming such trust decisions as part of the security decision making process helps to build the link between entities and expected behavior, which enables more accurate security decisions for mobile agent systems operating in open network environments. In doing so, trust-centric solutions contribute to the improvement of the overall security performance of the underlying system.

B. General Design Requirements for Trust Centric Solutions

We envisage that the design of trust-centric solutions can be achieved with the help of a new trust enhanced security architecture, where a trust management layer is constructed and

added transparently, on top of the conventional security layer. The trust management layer has mechanisms for managing the trust relationships and decisions. These trust-centric solutions should meet the following requirements: [1]

Requirement 1 - Trust Abstraction. Such solutions should be able to deal with the relevant types of trust information that are required in a trust enhanced security operation for mobile agents. They shall provide trust models capable of: 1) capturing all the fundamental trust assumptions, 2) modeling the trust relationships in a secure mobile agent system.

Requirement 2 - Trust Establishment. With the identified trust relationships, these solutions should provide mechanisms to support the following operations for trust establishment: 1) evaluating trust relationships to aid relevant security decisions, 2) deriving new trust relationships from existing ones using localised trust knowledge, 3) establishing trust relationships from internal sources (trust policy base), and external sources such as system assumptions and external trust knowledge by taking recommendations, 4) inferring and evolving trust relationships without prior interaction history and without recommendation, 5) Updating trust either through recommendation and direct experience at the end of transactions.

Requirement 3 - Integration of Trust and Security. The solutions should be designed to include a separate layer containing the new system components for the trust management functions, such as trust representation, trust recommendation, trust update, trust decisions and interfaces to security management layer.

V. Architectural Design Guidelines

A. The Overview

In designing the new architecture, we need to meet the design requirements of trust-centric solutions mentioned Section IV. The new trust enhanced security architecture has a trust management layer which is added transparently, on top of the conventional security layer. The trust management layer consists of new components and mechanisms such as the trust engine, trust base and associated trust management protocols. The protocols include trust enhanced authentication, authorization, trust recommendation and trust update.

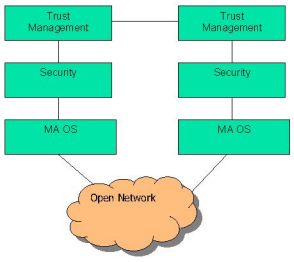

Fig. 1. A High Level View of ***MobileTrust***

Fig. 1 presents a high level conceptual view of the new trust enhanced security architecture (called ***MobileTrust***) for mobile agents. It has a three-layer structure above the open network environment, namely, the mobile agent operating system (MA-OS) layer, the security management layer and the trust management layer. Mobile agent systems and their operation in open networks have been covered extensively in other research works including [9], [5]. Our focus is on the security and trust management layers and their interworking relationships, which are the vital parts of the proposed architectural design of ***MobileTrust***. In Fig. 2 we show a more detailed conceptual view of the architecture showing the security and trust management components and their interactions in the context of trust enhanced security for mobile agent and host protection. Let us now outline the architectural design requirements:

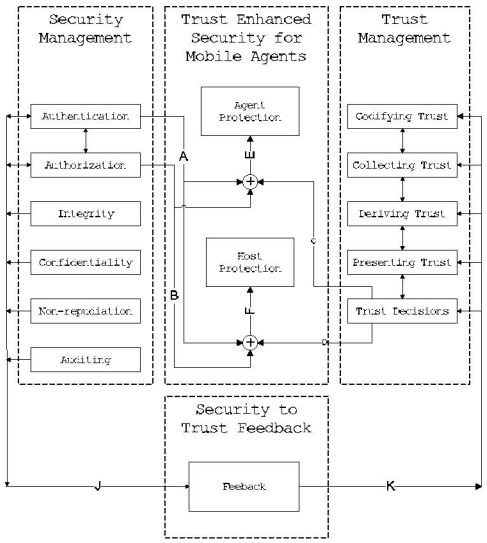

Fig. 2. A Detailed Conceptual View of the ***MobileTrust*** Architecture

B. Security Management

Security management in mobile agents plays an important role in protecting agents and hosts. There have been several research works in this area such as [6], [4], [3]. The following is a list of generic security management functions for mobile agents (see Fig. 2):

- *Authentication.* Before accepting an incoming agent, the executing host needs to know who its sender is. In this case, the receiving host needs to authenticate the agent. This process involves verifying the entity that developed the agent (i.e. the code author who programmed the agent), the entity that instantiated it (i.e. the owner host) and dispatched it to the current host, and the hosts who have actually executed the agent so far. Additionally, before sending an agent, an interim executing host needs to verify that the destination host is indeed the one it claims to be.
  - *Authentication of user.* The agent owner needs to authenticate himself or herself to a given executing host. A public key or symmetric key based encryption scheme can be used for the purpose.

[1] A range of trust models meeting the above requirements have been designed and fully documented in [8].

  - *Authentication of host.* Before a host starts to communicate with another host, it needs to know with whom it is communicating. This is important for ensuring the integrity and confidentiality in the communications with the hosts. Without this, we will not know from whom you are receiving an agent or to whom you are sending an agent.
  - *Authentication of code.* Before executing an incoming agent, the server needs to know who is responsible for the agent's implementation. Digital signatures are typically used for this purpose.
  - *Authentication of agent.* Before executing an incoming agent, the executing host needs to know who is responsible for this agent (i.e. who is the agent owner host).
- *Integrity.* To rely on an agent, one needs to make sure that no one has tampered with its state and code. Checking the integrity of the agent is the technique we can use to verify that no illegitimate alterations have been made to the state and code of an agent.
- *Confidentiality.* An agent may carry confidential (private) information that should be readable only by particular hosts or agents. Such information should be kept secret from other hosts and agents. In other words, an agent may be transported in a secure manner.
- *Authorization.* An incoming agent should only be given a right to access local resources according to its privileges. Authorization, or access control, provides mechanisms for specifying and enforcing an agent's capability to access information or to use a service provided by an executing host.
- *Non-repudiation.* This is concerned with ensuring that an agent or a host cannot deny that a given communication exchange or an action has taken place.
- *Auditing.* An auditing service records security related activities of an agent or a host for later inspection.

C. Trust Management

The trust-centric solutions involve a trust management layer. The operations of a trust management layer can be defined as the activities of *codifying, collecting, deriving* and *presenting* security related *trust evidence* for the purpose of making *evaluations* and *decisions* regarding trust relationships for mobile agent security. We list below a description of individual operations of trust management to be supported by the trust management layer in the new architecture:

- *Codifying Trust Evidence.* This refers to the process of expressing and presenting the security related trust relationships explicitly. As mentioned earlier in this section, we classify the trust into three classes in terms of the security contexts. We can further divide trust into two types in terms of source of generation, namely the *Hard Trust* and *Soft Trust*. *Hard trust* refers to evidence generated from security mechanisms, while *Soft Trust* refers to evidence generated from observable behaviors from the entities.
- *Collecting Trust Evidence* This refers to the process of generating trust evidence from one's own experiences, from system assumptions, or by taking recommendations from others in the system.
- *Deriving Trust Evidence.* This refers to the process of deriving trust evidence from combining trust evidence of one's own with that of recommendations from others.
- *Presenting Trust Evidence.* This refers to the process of presenting the trust evidence to both the human user via Graphical User Interfaces (GUIs) and to the host machines via a machine readable trust relationship data base (i.e. trust base).
- *Trust Evaluations and Decisions.* This is the process of actually using the trust evidence to make trust assessments and decisions, which will then form part of the security decision of the mobile agent system protection.

D. Interaction between Security and Trust

In our architectural model, we explicitly specify the interactions between the security and trust management layers. As shown in Fig. 2, these interactions are broken into two types which can be referred to as the "feed-forward" and feedback controls, following the formal terms of dynamical control systems [10]. The feed-forward controls determine how trust decisions are integrated into security decisions and the feedback controls provide trust update via feedback controls from the security operation results.

Feed-forward Controls. The feed-forward controls are used for trust enhanced security decisions and are described as follows: 1) *Feed-forward Control for Agent Protection.* The feed-forward elements in Fig. 2 are: arrow A carrying information from the security component, arrow C carrying information from the trust component and arrow E carrying information for the trust enhanced decision for agent protection. The *authentication trust* derived from security management component is used in arrow A and the *execution trust* is used in arrow C and the trust enhanced decisions (used in the trust enhanced itinerary composition to protect a mobile agent) are used in arrow E; 2) *Feed-forward Control for Host Protection.* The feed-forward elements are: arrow B carries information from the security component and D carries trust evaluation decisions from the trust component and F is the trust enhanced decision for host protection. Used in B is the *authentication trust* derived from the security management component and used in D is the *execution trust*. F provides the trust enhanced decision (used in the trust enhanced authorization for mobile agents to protect the underlying execution host).

Feedback for Trust Update. While the feed-forward controls are directly involved in the trust enhanced security decision making, the feedback controls between the security and trust components are responsible for harnessing trust related evidential information in the underlying system. The feedback controls are vital to the function of the feed-forward operations for updating the trust relationships which are the foundation of any trust management system. This differentiates our approach from existing social control based trust models, such as [11],

[12], [13], where the evidence used for trust evaluation is made solely from the observations of social behavior. The feedback is a generic concept. In our research, we restrict it to the activities of harnessing the relevant code and execution behaviors from the results of security operations such as authentication, authorization and integrality verification. Once harnessed, these information can then be used to map and update the trust model parameters for future decision making.

E. Implementation Notes

We have designed and implemented a Java based prototype of *MobileTrust* which implements all the trust management mechanisms discussed in the last section. We have also successfully integrated it with Aglet (a practical mobile agent platform from IBM). *MobileTrust* is fully documented in [8].

VI. Discussion and Analysis

In this section, we discussed how the trust enhanced architecture improves the mobile agent security decisions through the emergent properties of improved security performance in terms of agent and host protection. We will also give a brief introduction to the experimental results for validating such properties.

A. Properties of Trust Enhanced Security

Emergent Property of Trust Enhanced Agent Protection. Let us first consider agent protection. From a security management perspective, the agent protection level cannot be determined *a priori*. For example, while the agent owner host can initiate security protocols with other agent hosts, it may not have the knowledge of possible malicious behaviors of these hosts such as tampering the code, data, or staging a Denial of Service attack (DoS) (e.g. by refusing to receive the agent or by simply killing the agents). Furthermore because of the mobile agents' asynchronous mode of operation, the enforcement of the security protocols is not guaranteed. The successful completion of a mobile agent's itinerant tasks is dependent on other hosts' behavior and the security management functions cannot guarantee the behaviors of other hosts.

The following emergent property of the new architecture is used to address the above problem. This emergent property states that with trust management, an improved agent protection level can be assured *a priori*. This can be achieved by introducing a trust enhanced execution host selection process using a *trust enhanced itinerary composition protocol* [8]. This protocol uses trust evaluation to derive decisions on whether a particular host is trustworthy enough to be included into the mobile agent's itinerary. This is done by including a host into the itinerary if and only if the result of the trust evaluation of that host is over a certain trust level. This improves the probability of the expected benevolent behaviors from the selected execution hosts. This protocol is based on the probability principle that one can improve decision accuracy in the presence of uncertainty by making the probability of a favorable outcome as large as possible [14]. Experimental results outlined in the next section confirm this in Fig. 3 and Fig. 4.

Emergent Property of Trust Enhanced Host Protection. Let us now consider host protection by looking at the authorization of incoming agents. From the point of view of traditional security management, the authorization is of the incoming agent is determined based on the rights or credentials it carries [6], which are relatively static and do not guarantee how the agent will behave at runtime. In fact, when roaming in an open network, an agent's behavior is prone to change through potential tampering from the malicious hosts. The integrity checking and auditing mechanisms offered by traditional security management can only be used as post-event investigation tools. Just as in the case of agent protection, here, another emergent property of the new architecture is used to address the above problem. This emergent property states that with trust management, an improved host protection level can be assured *a priori*. This can be achieved by introducing *a trust enhanced authorization protocol* [8] to improve the probability of benevolent behavior of an incoming agent (or agents). The basic idea of the protocol is that a trust decision (based on code trust for the owner host and execution trust for the hosts which have executed the agent in the itinerary so far) needs to be made before the invocation of the authorization process. This additional step offers the opportunity to remove the malicious agents at runtime and hence improve the host security level. The experiment used to verify this property is described in [8].

B. Experimental Investigation and Analysis

To validate the architecture we have conducted an extensive experimental study [8]. We first integrate the *MobileTrust* into the *Aglets* platform to build an integrated trust enhanced security system for mobile agents. Then we establish the experimental framework for host and agent security evaluation by developing relevant metrics. Finally we developed system test cases and apply the evaluation framework to verify the emergent properties of the integrated system.

We have applied a standard empirical approach to study the characteristics of the *MobileTrust* architecture and to verify the emergent properties of improved security performance of the mobile agent system. Applying the *MobileTrust* prototype, we have conducted a wide range system tests to verify the security performance of the mobile agent system, for both agent and host protection. However, due to space constraint, here we only focus on presenting the results for the agent security performance; a full report of results is available in [8].

In testing the agent security performance, we have considered practical factors such as the population of malicious hosts and good hosts, and their behavior distributions in order to simulate realistic conditions for real applications. In contrast to the current practice where agents are dispatched into open networks without trust consideration, our architecture employs additional steps of trust decisions before furnishing the agent with a list of execution hosts (trusted itinerary).

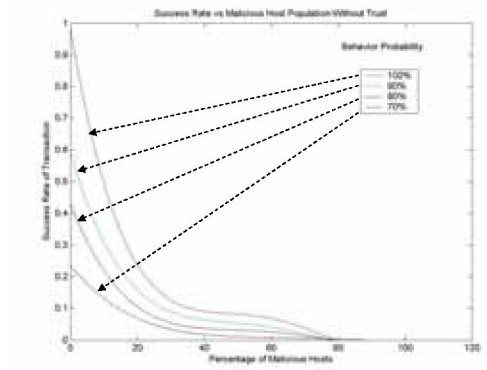

Fig. 3. Mobile Agent STR Without Trust Management

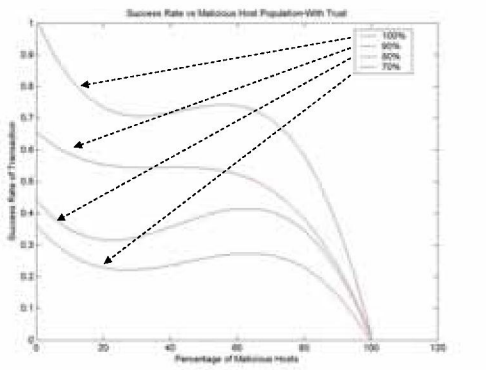

Fig. 4. Mobile Agent STR With Trust Management

Let us define three basic parameters developed for the evaluation of agent protection performance: PMH, BP and STR. PMH is defined as the Percentage of Malicious Hosts in the total population of hosts in a mobile agent system; BP is the Behavior Probability representing both the operational reliability of a good host and probability of malicious acts (characterized by data or code tampering) by a malicious host. STR, defined as Successful Transaction Rate, is the probability of having a successful computation by the mobile agent (i.e. no malicious tampering has occurred during the itinerant computation). STR is calculated as the ratio of the number of successful mobile computations to the total number of mobile computations. STR is measured against the parameters of Percentage of Malicious Host (PMH) and Behavior Probability (BP).

Figs. 4 and 3 are experimental results of the agent security performance measured as the STR using a sample Aglet mobile agent application. Figs. 4 and 3 correspond to the system configurations with and without trust management layer enabled. The tests are done over 250 runs of itinerant mobile computations, with an agent roaming through 6 hosts for each run. From Fig. 3 one can see that without trust management a mobile agent system is very susceptible to the presence of malicious hosts - the STR falls quickly with the increased presence of malicious hosts, a situation security mechanisms can not prevent. On the other hand, with the trust management, the agent security performance has improved significantly - the STR is able to withstand the impact of a relatively large presence of malicious hosts (see Fig. 4). This improved performance comes from the trust management architecture which integrates the trust decision with security enforcing the idea of using trust evaluation to "weed out" the malicious hosts. These results confirm the important emergent property of trust enhanced agent protection performance which can not be achieved with traditional security mechanisms alone.

VII. Concluding Remarks

To the best of our knowledge, this is the first paper to develop a trust enhanced security philosophy for mobile agents. It calls for the integration of trust into security. This has advocated a paradigm shift from security-centric to trust-centric solutions, which enables better security decisions for agent and host protections in the face of uncertainty of both the identities and behavior of entities in a mobile agent system. Specifically, solutions can be applied to provide enhancements to primitive mobile agent security operations: itinerary composition (in the case of agent protection) and authorization (for host protection). In developing such solutions, we have outlined a high level conceptual design of a new architecture along with the architectural design requirements. We have also discussed the new emergent properties of enhanced security performance exhibited by the new architecture. Finally we have provided a compact introduction to the experimental investigations and the results for validating the emergent properties.

References

[1] D. B. Lange and M. Oshima, "Seven good reasons for mobile agents," *Communications of the ACM*, vol. 42, no. 3, pp. 88–89, 1999.

[2] D. M. Chess, "Security issues in mobile code systems," in *Mobile Agents and Security, Editor Vigna*, vol. LNCS1419. Springer-Verlag, 1998.

[3] K. Schelderup and J. Olnes, "Mobile agent security - issues and directions," *In Proceedings of the 6th International Conference on Intelligence and Services in Networks, Barcelona, Spain*, April 1999.

[4] V. Varadharajan, "Security enhanced mobile agents," *Proc. of 7th ACM Conference on Computer and Communication Security*, 2000.

[5] D. B. Lange and M. Oshima, *Programming and Deploying Java Mobile Agents with Aglets*. Addison-Wesley, 1998.

[6] W. Jansen, "Mobile agents and security," *NIST*, 1999.

[7] E. Bierman and E. Cloete, "Classification of malicious host threats in mobile agent computing," *SAICSIT '02: Proceedings of the 2002 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology*, pp. 141–148, 2002.

[8] C. Lin, "Trust enhanced security for mobile agents," Ph.D. dissertation, Macquarie University, August 2006.

[9] N. Karnik and A. Tripathi, "Security in ajanta mobile system," *Software Practice and Experience, John Wiley and Sons*, May 2000.

[10] F. Raven, *Automatic control engineering, Fifth edition*. McGraw-Hill, New York, 1998.

[11] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara, "Reputation systems: Facilitating trust in internet interactions," *Communications of the ACM*, vol. 43, no. 12, pp. 45–48, December 2000.

[12] B. Yu and M. P. Singh, "Distributed reputation management for electronic commerce," *First International Joint Conference on Autonomous Agents and Multiagent Systems, Bologna, Italy*, 2002.

[13] T. Grandison and M. Sloman, "Specifying and analysing trust for internet applications," *Second IFIP Conference on e-Commerce, e-Business, e-Government*, October 2002.

[14] R. V. Mises, *Mathematical Theory of Probability and Statistics*. New York: Academic Press, 1964.