

# Mobile Agent Protection Mechanisms, and the Trusted Agent Proxy Server (TAPS) Architecture

Michelangelo Giansiracusa  
Information Security Research Centre  
GPO Box 2434, Brisbane, QLD 4001 Australia  
Email: mic@isrc.qut.edu.au

22 February, 2003

**Abstract:** *Autonomous mobile agents have been purported as a promising new alternative to traditional distributed computing approaches, bringing many advantageous features. Notwithstanding this, mobile agent technology has yet to see wide deployment in open networks like the Internet. The lack of interoperability between agent systems has impaired such progress. Moreover, fears of security breaches by malicious agent platforms has been a chief contributing factor in the lack of capital investment in Internet agent technology. The attraction of the mobile agent paradigm is diminished if the agent owner cannot trust the results of its deployed agents. However, as in other distributed systems, appropriately applying traditional distributed systems security techniques and incorporating trusted third-parties can discourage and detect bad behaviour by remote systems. This paper presents the TAPS architecture for mitigating security risks from malicious agent platforms. The TAPS architecture uniquely adapts highly standardised network security techniques (appropriately in an agent framework) so that previously untrusted agent platforms are held more accountable. With a key objective being compliance to the Mobile Agent Facility specification, a large number of agent systems*

*can readily utilise the features of this security architecture. The TAPS architecture provides a 'middle-man' server that anonymises authenticated agent entities in agent itineraries. We believe that the TAPS architecture will be a substantially more robust approach for protecting mobile agents than existing mobile agent threat countermeasures which tend to be piecemeal and inflexible in nature.*

**Key words:** Mobile agents, protecting agents, Trusted Agent Proxy Server (TAPS) architecture, Mobile Agent Facility (MAF), agent system interoperability.

## 1 Introduction

For over a decade, mobile agents have been tagged as the next revolutionary leap in distributed computing technologies. To date, their adoption has been limited mainly to corporate Intranet environments, for tasks such as automating network management activities and enhancing industrial robotic applications. [22, 4] Hence, agent applications have been predominantly confined to execute in environments span-

ning only a single trust domain.

The predictions of large numbers of agents autonomously roaming the Internet transparently performing tasks on behalf of a user has yet to be realised. Chief among the reasons for this limited investment in, and deployment of, agent technology via the greater Internet has been the absence of a standard infrastructure for supporting agents and security concerns specific to agent environments.

Agent platforms inherit the risks of traditional distributed system paradigms, but mitigating these risks with appropriate countermeasures is a relatively well-understood problem. More pressing agent-specific security concerns arise from malicious host attacks, where such hosts have the opportunity to subvert the work of the agent.<sup>1</sup>

In e-commerce applications, the interests of agent owners and agent platform owners are not the same - the host has an (albeit unethical) incentive to steal or change data, or modify processing to influence the agent's decisions. Hence, the agent paradigm model falls down unless you can trust the agent's results. This has proven to be a considerably more challenging problem and, even though studies have been undertaken to investigate possible remedies, a solid solution is a long way from the point of reality.

In section 2, distributed system paradigm types referred to throughout this paper are defined.

A critical analysis of the leading countermeasures proposed in the literature to date for protecting agents against malicious agent platform behaviour is presented in section 3.

In section 4, an overview of security mecha-

---

<sup>1</sup>An *agent platform* is interchangeable terminology for a *host system* - meaning an Internet host which is willing to let authorised agents travel to and use its system to perform some actions.

nisms for protecting more traditional distributed paradigm applications is provided.

In section 5, the vital issue of trust in remote platform systems is discussed in more detail, and there is discussion on whether agent paradigm security concerns differ markedly from other distributed system paradigms.

In section 6, we introduce our conceptual offering of a security architecture incorporating proxy server<sup>2</sup> hosts for mitigating the trust concerns in agent systems. This Trusted Agent Proxy Server (TAPS) architecture builds on solid traditional distributed systems techniques and localises the trust domain - thereby significantly reducing the likely threat magnitude, whilst also making it less arduous to pinpoint agent platforms who behave with malicious intent. The business model of such an architecture is outside the scope of this paper's intentions.

Finally, in section 7, a summary of discussed issues and the paper's conclusions are presented.

## 2 Distributed System Paradigms

To soundly appreciate the nature and promise of agent systems, it is helpful to look at how agents have evolved in the history of distributed application systems. This list discusses the evolution of distributed computing paradigms, the last being agents [5]:

- Most distributed applications fit into the *client-server paradigm*: a client makes some request to a remote server; the remote server has stored knowledge on how to process certain queries and return a result to

---

<sup>2</sup>It is important to note that *proxy server* in this architecture is quite dissimilar to that of a proxy server (firewall) in traditional network security parla.

the client computer. In client-server applications, the program code (or application logic) is static on the server computer, and the processor (i.e. abstract machine carrying out and maintaining state) and resources (i.e. I/O channels) for computation are always supplied by the server.

- The *remote evaluation paradigm* extended the client-server concept of distributed applications by transporting program code from the local to the remote computer, for execution there.
- The *code on demand paradigm* further extended the aforementioned concepts, by transferring the application logic in the reverse direction (i.e. from the remote to the local computer), and providing the computational platform and resources locally to ascertain some result.
- The *mobile agent paradigm* is a further extension, in that not only is the application logic transferred between computers, but the application state can be transferred from one computer to another. It is this latter property of agents which facilitates them in working autonomously (on behalf of a user or some local originating process) to travel between one or more remote computers, performing some work on those respective computers as required, usually returning some summary or result to the originating home computer.

The autonomous nature of agents can free-up time for their owners to perform other useful work in parallel. However, this genuine agent promise will not be widely realised throughout the Internet unless significant agent-specific se-

curity concerns are resolved and a standard infrastructure for agents is agreed upon. In the next section, these limitations and work undertaken to date by the agent research community to address them are discussed in greater detail.

### 3 Analysis of Agent Security Initiatives

There seems to be a general consensus that agent systems are highly vulnerable to security breaches, and that countermeasures against security attacks in agent systems are non-trivial. This is especially true in ensuring agents complete their designated work according to specification and free of abuse from potentially malicious host systems. [13, 7, 17, 5, 11]

Although there have been many agent system architectures built, they differ markedly in terms of infrastructural constraints and protocols. This lack of interoperability has been a contributing factor in the disappointing marketplace adoption of agent technologies. [15] The Foundation for Intelligent Physical Agents (FIPA) [8], and the CORBA-centric Mobile Agent Facility (MAF) [9] specification are initiatives which at least start to address this disparity.

Unfortunately, when it comes to standardised agent security, the story is even more bleak. FIPA, to take a case in point, has released a couple of security specific standards (among over 90 standards total) but they are now tagged with obsolete status, meaning they are no longer supported by FIPA! [23, 8] In MAF-compliant agent systems, the implementation must use CORBA middleware security services to satisfy their security needs. [9]

The security directives in both agent standardisation efforts are geared primarily toward

protecting the agent platform. This is understandable - as it is a much easier to countermeasure those attack threats, and it is currently more practical to standardise this security aspect of the mobile agent paradigm - at least until a great deal more research is undertaken by the agent security community into protection of agents from malicious hosts.

A larger number of leading agent systems comply to the MAF specification [1]. Our TAPS architecture, discussed in section 6, will be infra-structurally MAF-compliant and utilise CORBA services where appropriate and necessary.

### 3.1 Desired Agent Protection Security Properties

An explanation for the desired security properties in mobile agent systems for better assuring an agents' relative safety will be given below. An analysis of the security property support in existing schemes is given in table 1 on page 8.

**Agent Data Privacy:** Sensitive agent data values are not disclosed to untrusted prying host platform entities, or other agents.

**Agent Data Integrity:** Agent data values are not manipulated for malicious intent. Moreover, the intended code logic (e.g. flow sequence) must not be altered.

**Agent Availability to Resources:** Host platforms which claim to be available for offering services to agents must not deny resources to authenticated agents.

**Host Platform Accountability:** Host platforms should be held responsible for their interactions with agents.

**Host Platform Authentication:** Agents must be able to readily verify a platform is who it purports itself to be.

**Agent Authentication:** Hosts and other

agents must be able to readily verify that agents are genuinely who they purport themselves to be.

**Agent Accountability:** Agents should be held responsible for their interactions with other agent system entities.

**Agent Migration Protection:** Agents should not be susceptible to plaintext eavesdropping attacks, nor should their integrity be broken, on transit from platform to platform.

**Host Platform Non-repudiation:** If a host commits to a digital agreement, contract, sale, or other such transaction - then it must be highly provable that such commitment was made.

**Agent Non-repudiation:** If an agent commits to a digital agreement, contract, sale, or other such transaction - then it must be highly provable that it made such a commitment.

**Agent Anonymity:** The real identity, or related agent owner information, should only be available to trusted administrative third-parties (e.g. for legislative reasons).

**Revocation of Privileges For All:** Agent system entities whose privileges have expired or been taken away must not be allowed to continue using those revoked old privileges.

**Access Control For All:** Use of all resources is controlled, so that only those permitted a particular capability may use it.

### 3.2 Agent Protection Against Malicious Hosts

Following, we give a concise summary on the state of the art countermeasures for agent protection against attacks by malicious agent platforms, as presented in the body of research literature. For each countermeasure, a high-level explanation is given, followed by some of the most obvious advantages (prefixed by a '+' charac-

ter point) and disadvantages (prefixed with a '-' character point) specific to each proposal.

---

Execution Tracing:

Each agent platform logs every action performed by an agent whilst it is working there; and a cryptographic hash/fingerprint of the summary of work performed in an agent's visit is submitted to an external entity (possibly the agent home platform or a trusted third party).[31]

+ Can be a deterrent against malicious platform behaviour.

- Agent platforms must maintain large, non-repudiable log files.

- A secure protocol must be used for transferring the cryptographic hashes to external entities.

- Time synchronisation across platforms is required.

- Detection process requires manual checking, or software at the external entity to automate an intelligent analysis, of the results from verified hash summaries.

- It is unclear what, if anything, prevents the agent platform from falsifying the execution trace.

- Requires the agent platform to know the identity of the agent, and possibly record other sensitive information about or relating to the agent.

Partial Result Encapsulation:

Detect tampering by encapsulating the results of an agent's actions, at each platform visited, for subsequent verification - conceptually similar to the purpose of Message Authentication Code (MACs) in cryptography.[32]

+ Partial results, like MACs, are cheaper to compute than digital signatures (though, obviously, providing different properties to digital signatures).

+ Can provide forward integrity in multi-hop itineraries.

- Would require a trusted third party to timestamp a digital fingerprint of the results for any legal assurance.

- Which results to encapsulate may not be immediately obvious.

- Does nothing to ensure agent privacy.

Environmental Key Generation:

An agent's enciphered code is unlocked randomly, by reliance on an environmental condition trigger (such as a search string being found). The environmental condition can be hidden through a one-way hash.[24]

+ Reading the agent's code cannot reveal the triggering message.

- Agent execution environments (e.g. Java based systems) typically disallow the dynamic creation of code at run-time for execution.

- The agent system could simply read the data code out upon the triggering of the environmental condition, instead of executing it.

Computing with Encrypted Functions:

The aim is to ensure mobile code privacy, such that:

Alice has an algorithm to compute a function  $f$ . Bob has an input  $x$  and is willing to compute  $f(x)$  for her, but Alice wants Bob to learn nothing substantial about  $f$ . Moreover, Bob should not need to interact with Alice during the computation of  $f(x)$ . [26]

+ Non-interactive.

- Cryptographic theory has yet to find encryption schemes for computing arbitrary functions (without substantial knowledge attainment of the function) in a non-interactive manner.

- Does not prevent a number of other attacks such as replay, or modification in multi-hop agent systems.

Reference States:

Another approach to detecting modification attacks, reference states are agent states that have been produced by non-attacking “reference” hosts. [12] An alternative to the execution tracing approach [31], this new protocol offers a model where the execution on one host is checked unconditionally and immediately on the next host, regardless of whether this host is trusted or untrusted.

+ The approach preserves the qualitative advantages of the mobile agent paradigm like asynchronous execution.

- Introduces two new problems: input to the execution session on one host cannot be held secret to a second host, and collaboration attacks of two consecutive hosts are possible.

- It is purported that the overhead needed for the protocol roughly doubles the cost of the mobile agent execution.

#### Cooperating Agents with Mutual Itinerary

##### Recording:

Decrease susceptibility of undetected malicious host attacks by distributing data and operations across mutually supporting agents with carefully selected itineraries, thereby minimising the likelihood of them being located/executed at colluding malicious hosts simultaneously.[25]

+ Hosts must collude to ensure substantially damaging attacks.

- Additional resources are consumed by replicated agent entities.

- Does not specifically address agent privacy.

- May be difficult to ascertain appropriate mutual itineraries.

#### Obfuscated Code/Time-Limited Blackbox

##### Security:

Use code mess-up algorithms to make unintelligible the agent platform’s immediate perception on the role and data of the agent, and (in the time-limited blackbox security approach) where

possible restrict the useful lifetime of the agent’s code and data.[10, 11]

+ Can be very effective for relatively short-term secrets.

+ No cryptographic keys or algorithms required.

- All messed-up code and variable names can eventually be reverse-engineered, regardless of the complexity of the obfuscation algorithms.

- No blackbox algorithms exist that work for arbitrary input data (only polynomials and rational functions).

- There is no formal model for determining the relative strength of code mess-up algorithms (or for quantifying the directly proportional useful protection period).

##### Keylets:

The partitioning of mobile agent code and state information into self-contained components is discussed as a means of addressing the code security aspect for mobile agents. In the approach, these components are encrypted using symmetric keys which are then made available to platforms that will host the mobile agent in a network. The distribution of keys to platforms are determined through the execution of a specific type of mobile code that is termed a *keylet*. [28]

+ The propagation of keys provides a possible mechanism to implement trust propagation as part of an overall trust model for an agent system.

+ Separates agent mission task logic from code security functionality.

- Assumes the support of an available public-key infrastructure for the dissemination of public keys (of all the platforms in the system).

- Partitioning requires a third party code producer who could supply the mobile agent as a template to the agent owner.

- A large number of transactions related to keylet movement increase agent mission time and com-

plexity, and agent platforms may not be willing to support the increased computation and network traffic arising from their use.

- Key revocation is not adequately addressed.

The Supervisor-Worker Framework:

The framework consists of a coordinating entity (the supervisor) and several independent entities (the workers). The workers move from host to host and try to finish the tasks given from the supervisor. The supervisor can be mobile itself, moving closer to the area its workers operate in. The only prerequisite is that the supervisor must exclusively visit secure trusted places.[2]

+ Security measures are an integral part of the framework's design, thus common security retrofitting problems do not arise.

+ Eavesdropping information and tampering the agent is no longer possible or does not reveal any confidential information.

- In the worst case when there are no trusted sites available for the supervisor to coordinate things from, the supervisor must reside and work from its home (agent owner's) platform - but, what if the home platform wishes to go into disconnected mode?

- Significant time in preparation, and non-trivial decision making, could well be needed when initialising the Constraint Manager for the supervisor agent, and/or in real-time, to ensure the safety of his workers (and ultimately the agent's mission). This is placing extra, quite probably, unwanted burden on the agent's designer and/or user.

- It may well be the case that, in advance, which host(s) - if any - the supervisor/user can trust is unknown.

Self-Protecting Agents:

NAI labs research is claiming, on one of its website pages, to be developing strong protection for mobile agents by combining three

core techniques into agents it describes as "self-protecting" [19]: (i) *Distributed Agent State*, conceptually similar to a merging of strategies from the agent code and state component division in the keylets approach and mutual co-operating agents approach. [28, 25]; (ii) *Obfuscation with Periodic Regeneration*, conceptually similar to the ideas in the time limited black-box security approach. [11]; and (iii) *Monitoring and Recovering*, conceptually this appears to be an extension of the mutual co-operating agents approach detailed in [25].

+ The three-pronged self-protecting agent approach initially seems promising in theory.

- Following correspondence with an acting manager for the self-protecting mobile agent project, the desired solution has been hindered by technical challenges (particularly non-trivial obfuscation). The challenges requires significant state-of-the-art advancements, which experts in the field have no timeframe on their feasibility.

Secret Splitting Remote Digital Signing:

A multi-agent model is utilised along with simple *secret splitting schemes* for signing with shares of a key carried by members of a group of agents co-operating to accomplish a single task without the necessity of reassembling the key. This approach is said to allow secure remote digital signature generation. [20]

+ Simple scheme using well-known multiplicative and additive properties of RSA.

- If one of the agents carrying a part of the key is compromised, the signature generation scheme fails.

- Does nothing to ensure agent code/data privacy.

- Significantly more communication and coordination (of agents) is needed compared to a single agent mission implementation.

	Agent Data Privacy	Agent Data Integrity	Agent Availability To Resources	Host Platform Accountability	Host Platform Authentication	Agent Authentication	Agent Accountability	Agent Migration Protection	Host Platform Non-repudiation	Agent Non-repudiation	Agent Anonymity	Revocation Of Privileges For All	Access Control For All
Execution tracing		Y	L	L	Y	Y	L	L	Y	L			
Partial result encapsulation		Y		L					L				
Environmental key generation	Y										L		
Computing with encrypted functions	L	L								L			
Reference states		Y	L	L	Y	L	L	L	L	L			
Cooperating agents itinerary recording		L	L	L	L			L	L				
Time limited blackbox security	L	L		L							L		
Keylets	L	L			L	L		L					L
Supervisor-Worker Framework	L	L											
Self-protecting agents	L	L	L	L	L			L					
Secret splitting remote digital signing									L	L	L		

**Table 1: Agent security properties within existing countermeasure approaches**

**Table Key:**

**L=Limited Support by Scheme**

**Y=Core Support by Scheme**

Table 1 on page 8 gives our opinionated analysis of the security properties offered by the state-of-the-art agent protection schemes reviewed above.

What is clear is that there is no panacea solution for totally overcoming all of the associated risks involving agents, nor is such an ideal going to become close to reality in the near future. The proposed measures in the literature are often plagued by a number of problems including inflexibility, non-compatibility with other security measures, and impractical complexity for lightweight mobile code.

In the next section, we look at how more traditional distributed systems are secured - followed by a discussion on the importance of being able to trust remote platforms, in the section after that one. These sections are an important precursor to what follows, because our proposed architecture for agents builds on such traditional distributed systems security techniques and related thoughts.

## 4 Traditional Distributed Systems Security

Java applets have proliferated the interest in local execution of non-trusted code written by other parties on the Internet. Specifically, they are an example of programs that fit into the code on demand paradigm form of applications defined in section 2. Java's security approach combines strong language-typing mechanisms, a bytecode verifier, and sandboxing for restricting access to the local machine (i.e. the local machine being abstractly disjoint from the Java Virtual Machine, which is Java's platform-independent interpreter). [27, 30]

Digitally signed Java applets (or any other

mobile code that is signed) provide the recipient host with a means of ensuring the code has come from a known entity, and no modifications have corrupted the code's integrity in transit. It is important to note that signed code does not guarantee the code is non-malicious or free from error. However, signatures may be nested and trusted third-parties may review the code for deviations from its purported specifications, or may simply - as is more common - certify that the code has been produced by a reputable organisation.

The remote evaluation paradigm form of applications can apply the same set of threat countermeasures as those utilised by the code on demand paradigm form of applications since, from a general perspective, the risks faced by local execution of remote code are identical. Hence, given the reverse distinction of which host is acting in the local host role and which host is acting in the remote host role, the threats and their sources are semantically equivalent.

Security for more general client-server applications is relatively well understood. End-to-end cryptographic technologies like Secure Sockets Layer (SSL) and Secure SHell (SSH) operate at the session layer. Public-key technologies and public-key infrastructures, properly utilised, can enhance enterprise security - providing a large number of security services including confidentiality and integrity of data, entity authentication, and non-repudiation of data transmission and receipt.

Distributed systems security services have been packaged into a number of proprietary and non-commercial distributions. Some of the more popular distributed system security software distributions include Kerberos [18], SESAME [6], DCE [29], Windows 2000 Server [16], and CORBASEC [14].

So, it can be seen that traditional distributed systems security techniques are relatively well-understood and widely implemented, especially when compared to the lack of effective agent-specific countermeasures for mitigating the threats stemming from malicious agent platform attacks.

Nevertheless, there is a significantly large assumption in all forms of remote processing systems that the remote system can be entrusted to not only perform in an ethically sound manner, but to handle our sensitive data professionally and securely. The next section elaborates on why trust issues require serious contemplation when determining security baseline requirements in designing distributed architectures, distributed application systems, and why this is further exacerbated in the mobile agent paradigm - the most flexible of the distributed system paradigm types, and thus facing the most serious security threats.

## 5 Trust Issues in Distributed Platforms and Execution Environments

The whole notion of trust and trusted systems is arguably one of the most misunderstood subjects in IT security. Trust, however, is undeniably a critical requirement in bridging sufficient confidence in the processing results from the variety of distributed system paradigms and available remote platforms, not the least in the case of agent systems spanning the greater Internet.

Despite the existence of standardised evaluation and certification criteria for computer systems assurance (e.g. TCSEC/Orange Book, ITSEC, and the Common Criteria), there is arguably no workable benchmark when it comes

to the security of networked and distributed systems, especially when they go beyond a small and well-defined trusted computing base (TCB). The problem is accentuated by the size and complexity of today's network operating systems which are often susceptible to widely published exploit attack scripts. [21]

The Trusted Computing Platform Alliance (TCPA) is an initiative which could bring promise for increasing trust in a remote platform. A tamper-resistant processor integrated with the motherboard checks the ROM. The TCPA goal is to use trusted hardware as an anchor point from which to establish a chain of trusted components. Hence, verify component 1 from hardware - if OK, use component 1 to verify component 2 - if OK, use component 2 to verify component 3, and so on. [3] For such technology to become viable, large (e.g. public-key) infrastructural and business changes will have to be seen. Moreover, the proposed scheme will raise a number of challenging problems, including: issues with key revocation, how to recover from keying failures integrated into the hardware, and scalability and performance limits.

### 5.1 Trust in Distributed Systems

Well, that is where things are now and where they are likely to be heading in terms of trusted systems, but what about trust in traditional client-server systems (and other traditional forms of distributed applications) and the difference to that of the risks in agent systems?

Why do we trust servers at all with our personal details? Do we have any sound and/or verifiable knowledge of their credibility and assurance levels? If there is security present in a distributed system it is often limited to end-to-end session level encryption, such as that provided

by SSL. What is, at least, as worrying is what happens to our private details such as credit card details once our data leaves that secured virtual channel.

There may well be a solid line of argument saying that, in traditional forms of distributed systems, trust can be based on brand recognition and reputation. Furthermore, trust domains are readily more comprehensible.

In practical terms, however, the fundamental difference in mobile agent systems to those of more traditional distributed systems is that, by definition, data and executable code may travel to more than one agent platform. With visits to additional platforms, there comes a rising risk association that one or more visited agent platforms may digress from acceptable ethical practices.

Notwithstanding this, however, we believe that in a well-designed agent architecture it is possible to significantly reduce this risk association by appropriately and uniquely applying traditional distributed system security concepts (and improving on agent protection proposals, which can be plugged into the architecture on availability over time). An introduction to our proposed security architecture for realising this risk reduction follows in the next section.

## 6 Proposed Security Architecture

Without operating on a trusted hardware and software platform, there is no way to succinctly verify the assurance of server environments. Agent platforms, in any non-trivial sized system, must have sufficiently high quantities of memory and processor speed. This rules out smartcards as being a viable option for encapsulating agent

platforms.

The methodology of uniting TCPA and a secure operating system kernel to establish a tamper-resistant boot record of loaded trusted components appears to be a sound approach for ascertaining applicable trust in mainstream systems and an agent's integrity. Notwithstanding this, however, such technology has first to overcome a number of ensuing political and legal battles - as well as the aforementioned practical dilemmas - before being globally standard in corporate and right across the range of mainstream personal computing hardware and operating system software platforms. [3]

The proposed agent architecture introduced below in some detail is based on the idea of a trusted proxy server host, which acts as a strong deterrent against non-ethical behaviour from potentially malicious agent platforms. Localising trust in the proxy server host reduces the accumulative trust needed in remote hosts. Moreover, it downsizes the headaches in identifying agent platforms that behave with malicious intent.

The proxy server hosts, interfacing directly with agents and agent platforms, will have numerous important responsibilities and properties, including (but not limited to) those which are outlined below.

### **Agent Authenticator:**

Only readily identifiable and well-known TAPS architectural entities will be permitted in the system. To this end, a Public-Key Infrastructure (PKI) will be utilised by the TAPS for authentication of agents, agent home systems, and agent platforms. Authentication responsibility is delegated to the TAPS Authentication Module.

### **Services Query Centre and Agent Router:**

The incoming agent, unless an explicit itinerary of agent platforms to visit is expressed, should

be able to query the TAPS on which suitable host(s) to visit in the pursuit of fulfilling its work requirements. Regardless of an explicit pre-existing itinerary or a locally-generated itinerary, the TAPS will route the agent to the agent platform next in the agent's itinerary. If the agent has requested an execution trace-like service, then the agent should be re-routed back to the TAPS in-between each visit to an agent platform. Service querying and agent routing will be handled by the appropriate CORBA services.

**Agent Anonomiser:**

Agent platforms will never see an agent's long-lived tied private key when agents visit. They will, if the agent requires it, only be able to see a random short-lived (one hop useful only) secret session key. In other words, if there is employment of cryptographic primitives requiring a key when an agent is on an agent platform then the agent will use the anonymous session key generated by the Session Key Distributor (SKD) (Key Distribution Centre (KDC)-like) portion of the proxy server host before the agent is routed to its next agent platform. Moreover, the agent platform should not be able to identify the agent owner (from spying out the agent code, for example) because the agent's public keying material including the corresponding certificate has been stripped from the agent packaging as the agent has already been identified as an authenticated agent entity at the proxy server host before routing it to the next target agent platform. The anomomisation process is the responsibility of the TAPS Authorisation Module, which interacts with the TAPS Privilege and Accountability Database (PAD) and the SKD.

**Privilege Management Resources Verifier and Maintainer:**

Legitimate access to agent platform resources

will be controlled primarily by Role-Based Access Control (RBAC), probably modelled within X.509 Authorisation Privilege Attribute Certificate (PACs). Besides typical access control resource allowances (e.g. disk file permissions), agent specific allowances will also be modelled (e.g. maximum CPU resource usage, time-to-live, etc.). Agent platforms will *push* (upload) updated privilege records to the TAPS on a periodic batch basis, but because RBAC management is employed these updates will be kept to a minimum.

**Secrets Maintainer:**

Sensitive agent data which should not be privy to specific agent platforms can be encrypted and stored in the PAC or simply left at the proxy server host until needed. It may be that certain agent owners distrust specific agent platforms and the proxy server host should treat this knowledge as private and take it into account when preparing to dispatch the agent to the next target agent platform. In effect, the proxy server host acts like an intermediate client - in a client-server sense - for every server (i.e. agent platform) in an agent's itinerary (see the conceptualised diagram, figure 1 on page 13 below). Therefore, the risk levels can be reduced to that associated with traditional forms of distributed applications - in particular, a fusion of the client-server and remote evaluation paradigms. Critically, agent platforms in this architecture are faced with a strong deterrent against behaving maliciously because tracking of anomalies can simply be reduced to a single hop visit from the proxy server host.

**Minimal System with MAC Provisions:**

Both agent owners and agent platforms require the TAPS to be highly trusted, and to be unbiased in its routing decisions. The TAPS must have a cut-down, stable configuration. Manda-

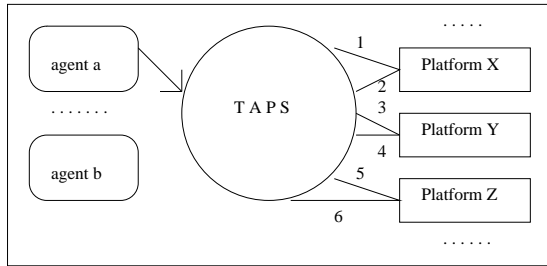


Figure 1: *TAPS conceptualised, with an agent passing through for processing on a 3-hop itinerary.*

tory Access Control (MAC) provisions will be applied to the TAPS to strengthen confidentiality, integrity, and reliability features (and in the process also adhering to the principle of least privilege).

#### **Non-Repudiable Auditor:**

Audit and logging is critical to the proxy server host's purpose - not just to track down errors in processing, but most importantly for the correct resolution of liability disputes in the case of security breaches.

#### **Secure Hub:**

The TAPS can act as a secure repository for when the agent has finished its mission, but the home platform is offline (i.e. is in disconnected mode) and thus the agent's results cannot be returned immediately. As computing hardware downsizes physically and users become increasingly more mobile, such a scenario is likely to become common.

#### **Agent System Independent:**

*Because no agent execution actually takes place on any TAPS, any agent system can choose to adopt and utilise security features intrinsic to the TAPS architecture. CORBA Mobile Agent Facility (MAF) specification compliance will be a key feature of the TAPS architecture.*

#### **Highly Time Sensitive and Replicated:**

All critical clocks in the agent system must be synched, and the proxy server hosts must have high uptimes and be replicated and consistent.

#### **Standards Based:**

Whenever deemed appropriate and feasible, accepted best industry standards will be used in the design and implementation of the TAPS architectural solution.

### **6.1 Example Scenario**

Considering figure 1, if *agent a* is deemed to be an authenticated agent entity by the independent TAPS host, the TAPS host will check the agent's *request ticket* to determine a required agent itinerary and detail appropriate authorised role privileges for the agent to complete its mission. All public keying and origin identifying information, for example the agent's digital certificate and source IP address is removed (and replaced, for example, by the TAPS host's IP address). The TAPS host also digitally signs the delegated roles of the agent as permitted on the target agent platform(s), encapsulates this in a Privilege Attribute Certificate (PAC), and forwards this PAC and the anonymised agent to the next end server (agent platform) machine.

### **6.2 Mobile Agent Facility Compliance**

Mobile agent systems interoperability is fundamental if mobile agents are to be a truly core technology in the Internet and in an electronic services marketplace. The CORBA-centric Mobile Agent Facility (MAF) specification [9] provides solid abstractions for agent system interoperability. Because agent technology is still in a period of infancy, MAF is neither heavyweight nor stringent. It provides a collection of defi-

nitions and interfaces that are both simple and generic, allowing for future advances in mobile agent technology.

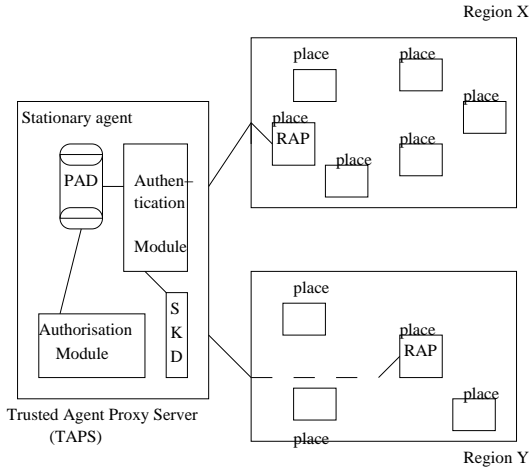


Figure 2: *TAPS major components, and MAF regions with Region Access Points (RAP)*

The two software abstractions MAF stipulates for compliance are the *MAFAgentSystem* and *MAFFinder* interfaces.

Management of mobile agents throughout their life cycle is the responsibility of the *MAFAgentSystem* interface - a compliant implementation will provide method implementations for `create_agent`, `fetch_class` in the case of a non-object oriented agent system, `find_nearby_agent_system_of_profile`, `get_agent_status`, `get_agent_system_info`, `get_authinfo` for querying whether an agent was authenticated and how, `get_MAFFinder`, `list_all_agents`, `list_all_agents_of_authority`, `list_all_places`, `receive_agent`, `resume_agent`, `suspend_agent`, `terminate_agent`, and `terminate_agent_system`.

Because the CORBA services are designed for static objects, all cases for using the

CORBA naming service for agent location information may not be handled appropriately. The *MAFFinder* interface is declared to handle this shortcoming. The MAFFinder functions as the principal source for dynamic name resolution and the repository for the location of agents, places, and agent systems. A compliant implementation will provide method implementations for `lookup_agent`, `lookup_agent_system`, `lookup_place`, `register_agent`, `register_agent_system`, `register_place`, `unregister_agent`, `unregister_agent_system`, and `unregister_place`.

By definition, the MAF abstraction of a *place* is what we have been commonly referring to as an agent platform in this paper. Places are a context within an agent system in which an agent can execute. An agent can travel and do work at places sharing the same *agent profile*. A *region* is a set of agent systems that have the same administrative authority, but are not necessarily of the same agent system type. Having the necessary authorisation- agents, agent systems, and non-agent systems may communicate with any inter-region target via a region host which has been designated as a *region access point (RAP)*. Communication between agent systems and clients outside of a region and entities within the region via this region access point is similar to the scenario of firewall filtering when an untrusted Internet entity attempts communication with an entity located within a corporate Intranet domain having a different trust level.

Figure 2 on page 14 shows some of the more major MAF physical component abstractions and their high-level relationship with the planned TAPS. The TAPS host will be abstracted as a *stationary agent*, which the MAF specification makes provision for. The TAPS architecture will be CORBA Common Secure In-

teroperability Specification (CSI) level-2 compliant. CSI level 2 provides for the strongest security provisions including integrity, confidentiality, replay detection, misordering detection, full delegation of privileges, and mutual authentication.

## 7 Conclusion

Mobile agent systems form the latest evolution in distributed paradigms. The secure Trusted Agent Proxy Server (TAPS) architecture introduced in this paper can be seen as an agent architecture which is a hybrid extension of client-server systems and remote evaluation systems.

Although existing countermeasures are relatively well-defined and accepted for mitigating the risks of malicious attacks in these more traditional distributed platforms, securing agents has been bracketed as a significantly more challenging exercise.

In this paper, the leading research offerings for protecting agents against malicious hosts have been overviewed. In particular, it can be concluded that whilst the existing countermeasure approaches offer preliminary attempts at downsizing some of the risks in agent systems, they each suffer from a number of drawbacks. Common weaknesses include inflexibility, lack of robustness, concerns with multiple trust domain issues, and high computational complexity for lightweight mobile code.

We introduced the conceptual offering of a trusted proxy server host as an architectural approach to lowering the risk concerns of agent code executing on potentially non-trusted remote hosts. A large number of security services can be provided by the proxy server host, including authentication of entities, anomomisation of

agent owners, a trusted time server which can be synched into the wider agent infrastructure, non-repudiable logging of actions and/or results, data secrets maintainer, Role-Based Access Control management, and unbiased secure routing.

The proxy server host architecture reduces an agent's risk concerns in a number of ways, including - but not limited to: (1) storing longer-term private keys (and possibly other secrets) at the proxy server host between hops to agent platforms, (2) securely maintaining a non-repudiable record of results from previous agent platform visits in the agent's itinerary, and (3) in using role-based access control and only allowing the forwarding of authenticated agents to agent platforms, maliciously-motivated agent platforms should not be able to readily identify specific agents to target for attacks.

The mobile agent security architecture research project is still in its early stages, but the plan over the next two years is to further design the proposed TAPS architecture, build a TAPS server and proof-of-concept application, and release the work to the greater agent security research community as a practical high-level solution for bringing greater security to agents whilst keeping such a solution agent-system independent and highly interoperable. The contribution will uniquely draw inspiration from proven traditional distributed security mechanisms, agent system standards, and utility existing agent protection mechanisms.

## References

- [1] Mobile Agent Platform Assessment Report. Retrieved July 26, 2002, from <http://citeseer.nj.nec.com/352727.html>.
- [2] Building Secure Mobile Agents: The Supervisor-Worker Framework, Febru-

- ary 2000. Retrieved April 25, 2002, from <http://citeseer.nj.nec.com/499071.html>.
- [3] Ross Anderson. TCPA/Palladium Frequently Asked Questions, 2002. Retrieved July 27, 2002, from <http://www.cl.cam.ac.uk/~rja14/tcpa-faq.html>.
- [4] P. Bellavista, A. Corradi, C. Stefanelli, and F. Tarantino. Mobile Agents for Web-based Systems Management. *Internet Research*, 9(5):360–371, November 1999.
- [5] Anthony Chan and Michael Lyu. Chapter: The mobile code paradigm and its security issues, 1999. Retrieved April 26, 2002, from <http://citeseer.nj.nec.com/421754.html>.
- [6] Joris Claessens. SESAME, 2000. Retrieved August 19, 2002, from <http://www.cosic.esat.kuleuven.ac.be/sesame/>.
- [7] William M. Farmer, Joshua D. Guttman, and Vipin Swarup. Security for Mobile Agents: Issues and Requirements, 1996. Retrieved June 16, 2002, from <http://csrc.nist.gov/nissc/1996/papers/NISSC96/paper033/SWARUP96.PDF>.
- [8] Foundation for Intelligent Physical Agents. About FIPA, 2002. Retrieved August 25, 2002, from <http://www.fipa.org/about/index.html>.
- [9] GMD FOKUS and IBM. Mobile Agent Facility Specification V1.0, 2000. Retrieved September 25, 2002, from <http://cgi.omg.org/docs/formal/00-01-02.pdf>.
- [10] Fritz Hohl. An Approach to Solve the Problem of Malicious Hosts. Technical Report 1997/03, Universitt Stuttgart, March 1997.
- [11] Fritz Hohl. Time Limited Blackbox Security: Protecting Mobile Agents From Malicious Hosts. *Lecture Notes in Computer Science*, 1419:92–113, 1998.
- [12] Fritz Hohl. A Protocol to Detect Malicious Hosts Attacks by Using Reference States. Technical report, Universitt Stuttgart, Fakultt Informatik, Technical Report Nr. 1999/09 1999.
- [13] Wayne Jansen. Countermeasures for Mobile Agent Security. *Computer Communications*, page Special Issue on Advanced Security Techniques for Network Protection, 2000.
- [14] Konstantin Beznosov. CORBASEC Frequently Asked Questions and Answers, 1999.
- [15] Ming Kin Lai. Standards for Agents: MASIF and FIPA Specifications, 2001. Retrieved July 20, 2002, from <http://www1.ics.uci.edu/~mingl/agent.html>.
- [16] Microsoft Corporation. IT Introduction to Windows 2000 Security, 2002. Retrieved August 19, 2002, from <http://www.microsoft.com/windows2000/server/evaluation/business/secintro.asp>.
- [17] Kari Miettinen. Security Issues in Agent Technology, December 1998. Retrieved April 24, 2002, from <http://www.cs.helsinki.fi/u/kraatika/Courses/Agents/miettinen.html>.
- [18] MIT. Kerberos: The Network Authentication Protocol, 2002. Retrieved August 19, 2002, from <http://web.mit.edu/kerberos/www/>.
- [19] NAI Labs. Secure Execution Environments: Self-Protecting Mobile Agents. Retrieved August 17, 2002, from <http://www.pgp.com/research/nailabs/secure-execution/self-protecting.asp>.
- [20] O. Kaan Onbilger, Randy Chow, and Richard Newman. Remote Digital Signing with Mobile Agents. In , 2002.
- [21] Rolf Oppliger. Trouble ahead, trouble behind: The future of computer security. *Computer Security Journal*, 17(1):19–25, 2001.
- [22] H. Parunak. Practical and Industrial Applications of Agent-Based Systems, 1998. Retrieved April 25, 2002, from <http://www.cs.umbc.edu/agents/>.
- [23] Stefan Poslad and Monique Calisti. Towards improved trust and security in FIPA agent platforms. In *Autonomous Agents 2000 Workshop*

*on Deception, Fraud and Trust in Agent Societies*, Barcelona, Spain, 2000.

- [24] James Riordan and Bruce Schneier. Environmental Key Generation towards Clueless Agents. *Lecture Notes in Computer Science*, 1419:15–24, 1998.
- [25] Volker Roth. Mutual Protection of Co-operating Agents. In *Secure Internet Programming*, pages 275–285, 1999. Retrieved June 26, 2002, from <http://citeseer.nj.nec.com/roth99mutual.html>.
- [26] Tomas Sander and Christian F. Tschudin. Protecting Mobile Agents Against Malicious Hosts. In Giovanni Vigna, editor, *Mobile Agents and Security, LNCS*, pages 44–60, Heidelberg, Germany, 1998. Springer-Verlag.
- [27] Sun Microsystems Corporation. Frequently Asked Questions - Applet Security, 2002. Retrieved August 19, 2002, from <http://java.sun.com/sfaq/>.
- [28] Hock Kim Tan and Luc Moreau. Mobile Code for Key Propagation. Retrieved June 2, 2002, from <http://citeseer.nj.nec.com/458860.html>.
- [29] The Open Group. What is Distributed Computing and DCE?, 2002. Retrieved April 29, 2002, from <http://www.opengroup.org/dce/>.
- [30] Bill Venners. The lean, mean, virtual machine: An introduction to the basic structure and functionality of the Java Virtual Machine. *Java-World*, June 1996. Retrieved April 23, 2002, from [http://www.javaworld.com/javaworld/jw-06-1996/jw-06-vm\\_p.html](http://www.javaworld.com/javaworld/jw-06-1996/jw-06-vm_p.html).
- [31] G. Vigna. Protecting Mobile Agents through Tracing. In *Third Workshop on Mobile Object Systems*, 1997. Retrieved July 2, 2002, from <http://citeseer.nj.nec.com/vigna97protecting.html>.
- [32] Bennet S. Yee. A Sanctuary for Mobile Agents. In *Secure Internet Programming*, pages 261–273, 1999. Retrieved May 20, 2002, from <http://citeseer.nj.nec.com/yee97sanctuary.html>.