Extending the Buddy Model to Secure Variable Sized Multi Agent Communities

John Page, Arkady Zaslavsky, Maria Indrawan

School of Computer Science & Software Engineering, Monash University, Melbourne
(john.page, arkady.zaslavsky, maria.indrawan)@infotech.monash.edu.au

Abstract: This paper describes an extension of the Buddy model of agent security. The Buddy model provides a security layer that encapsulates all agents within the multi-agent community and therefore, extends a security cover to all agents of the community. While the current model is able to service several scenarios involving fixed sized agent communities, it does not consider variable sized agent communities. This paper describes an extension to the existing model that covers this aspect. The proposed extension increases the application coverage and flexibility of the Buddy model and makes it a potent security mechanism for multi-agent communities. Further, the paper establishes that agent migration does not lead to any schema operating overheads; on the contrary it contributes to maintaining an effective level of performance integrity within the schema.

Keywords: Agents, Mobility, Security, Community, Buddy

Introduction

The popularity of web-based services using portable languages such as Java [4] and scripting languages such as TCL/TK [14] encouraged the development of the Mobile Agent (MA) paradigm. To describe the motivation behind the MA paradigm, researchers and developers have given several definitions. Some of these definitions have introduced new terms causing confusion between various forms of mobile code in use. Franklin & Graesser [7] have produced an interesting taxonomy of different agent architectures that were evolving around the time that Java was establishing itself as a web-based application development solution. Their study identified and summarised the various characteristics that were seen to form a part of agent behaviour.

Using the commonality of these characteristics, [7] proposed a definition that identified an agent as an autonomous system situated within an environment on which the agent continuously acted over time, so as to effect its future sensing. While there are several agent based toolkits [1] which are being used for educational as well as commercial application development, some of the early toolkits such as Aglets [8], Odyssey [2], Grasshopper [3], and Concordia [13] are still popular as interesting
sources of research for the agent paradigm. Using these toolkits, researchers have attempted to explore the somewhat complex dynamics of agent oriented software engineering and propose theories for the further evolution of the paradigm. One much researched area of agent oriented computing is security. The need for entities to interact with a minimum level of mutual trust is a well appreciated and much sought after goal.

While the presence of mobility has made the nature of agent oriented applications highly pervasive, it has also compromised the achievable level of security within the applications. Agent oriented transactions are highly susceptible to malicious tamper attacks [15] and the integrity of the transactions taking place can sometimes be questionable. To achieve an acceptable level of security, researchers have proposed various methods, some of which have been summarised by [9]. While the many theories and methods proposed by researchers may be effective to some extent, there is no clear single solution to beat the challenges posed by the security threats to the MA operation.

This paper examines the reasons behind the lack of a secure computing infrastructure for mobile agents. Further it summarises some of the possible threats faced by MAs from malicious entities. To counter these threats, the paper describes the Buddy model schema of agent security [5] and presents the schema operation as an effective measure of assessing threat levels in a distributed multi-agent community scenario. The paper discusses how mobility, commonly regarded as a cause for an increasing level of security vulnerability, is used by the Buddy model in enforcing greater reliability. The schema also promotes operating transparency between the MA and the agent server which is an effective trust building mechanism.

This paper is organised as follows. The section “Barriers to a Security Operation” describe the various factors that have prevented proposed security measures from being effective within mobile agent systems. This section also summarises some of the possible security threats faced by MAs. Further it also describes the relevance of the Buddy model schema in providing a security mechanism in distributed systems. “Buddy Model Schema Architecture” describes the architecture and the operation of the Buddy model. “Buddy Model Schema Operation” describes the implementation of the system and presents an analytical view in discussing agent migration and its effect on the schema’s operation. “Related Work” describes related work and compares the advantages of the schema by comparing it with other proposed agent security schemas. “Conclusion and Future Work” concludes the paper with a summary of the schema operation and the results presented in this paper. The section also provides a brief indication of future work. Henceforth for the remainder of the discussion the term agent and MA shall be used synonymously.

Barriers to a Security Schema Operation

MAs enable the development of pervasive applications. They can migrate from one system to another carrying with them information obtained along its travels. While this feature allows the agent operation to be highly flexible and gives the MA a chance to make the most optimal decisions with respect to its future execution, it also
makes the MA susceptible to malicious attacks that target its code and data. Security proposals for the MA become dust in the wind, if the host Mobile Agent Server (MAS) is malicious. The main drawback that prevents security measures from being successful is that the agent host has control over the execution environment. This allows it to inspect the agent code prior to executing it, thus allowing it to pre-empt the security trigger from firing within the agent. Thus, the MAS can bypass the agent security mechanism allowing it to compromise the agent at will.

A security mechanism that allowed the agent to generate its security mechanism at runtime could possibly have been successful but since the agent host possesses a copy of the serialized agent code, the agent host could replay the agent in a controlled environment and study the execution of the run-time security generation. Thus, when the agent would actually execute in the ‘real’ time environment, the agent server would not allow the security trigger to fire.

While in the past, some security proposals have involved cryptographic mechanisms, the argument against them becoming the de facto solution is a strong one too. Cryptographic mechanisms can be rendered useless if the encryption key employed is compromised. A further argument against the use of cryptographic techniques is the expense in the processing involved. For example, consider an agent operation that normally would use up five processing cycles for its business function. For its cryptographic function, the agent might need to use another five processing cycles. This will destroy the viability of the agent function as far as cost vs. benefit analysis is concerned.

The use of cryptographic techniques also creates a level of uncertainty for the MAS. For example, consider the case where in the agent host is not intending a malicious action on the arriving agent and is interested in providing the agent with its requested resources. If an arriving agent carries along with it a host of cryptographic mechanisms, it becomes difficult for the agent server to gauge the intentions of the agent. In such situations, it is even possible that the server may regard the agent as a malicious entity with an unclear agenda and deny the agent execution privileges. While the prevailing scenario does not encourage a solution that empowers an agent to take control of the execution environment, as this approach would be unacceptable to the agent server, the question of maintaining the integrity of the serialized agent code and the executing instance of the agent remains largely unanswered.

Agent community operations bring in an additional level of complexity to the already vulnerable state of the agent security dimension. Communities involve collaboration, communication and in the case of mobile communities, there is migration also that needs to be considered. In order to safeguard the interests of the community, the community operations require a critical level of security to be in place at all times as a failure at any one level could compromise the entire agent operation.

The Buddy Model [5, 10, 11] proposed a security schema for the operation of mobile agents while operating as a part of a community. While [10] laid down the rules governing the architecture, [11] demonstrated the schema’s ability to counter security threats in mobile business scenarios. The ability of the schema to involve all members of the agent community in the security function enabled a cost effective solution that spread the risk of failure equally among the community members. According to the Buddy model, the security function of each MA is backed up by two other agents,
referred to as Buddies. This implies that in order to compromise any one agent within the community, a malicious attacker has to attack and compromise at least two other agents, which are located on different physical machines. Thus, the difficulty level of compromising the schema is inordinately high. This factor deters malicious programmers making the schema a viable option for the protection of mobile agent communities.

Currently MAs are being used to develop web-based applications [1]. In such scenarios, security becomes a critical requirement. Most mobile agent toolkits implement a two-pronged security defence. They rely on authentication and authorization of agents to filter out the genuine agents from the rogue agents [6]. As discussed in [10], while this approach provides the basic level of application based security, it does not guarantee the protection of the system from hostile action as malicious entities can still slip through using forged or stolen credentials. Thus, there is an ever present need for MAS to employ a security mechanism that functions well within the inner reach of the system and continuously monitors the key points of the system for tampering attacks. Further more, the authentication and authorization approach aims to protect the MAS from malicious agents but it does not contribute towards extending the security cover to MAs. This leaves MAs vulnerable to the malicious attacks of the host servers.

The Buddy model of security counters this drawback by providing a security cover that continuously monitors the MAs. To determine the level of security within the system, the schema provides a Confidence Factor (CF) function. This function can at different instances in time, report the level of security present in the system in tangible terms. The next section describes the operation and the proposed extension to the existing model.

Buddy Model Schema Architecture

While the architecture and the different configuration models that evolve from the schema have been discussed with detail in [10], this section offers a brief recap. Before going into the schema operation, it would be useful to understand some terminology used in the schema operation and to briefly recapitulate some of the rules that govern the operation of the schema and reduce the risk factor of a hostile action on the community members. Each cycle in the security schema has two phases. Further, the contribution of each member of the community to the security function is equal. In return all members are covered by the security schema over the two schemas.

Agents within the Buddy model have a role based action. The schema identifies two roles, namely Buddy agents and Protected Agents (PAs). PAs are the agents that are being monitored during a phase while Buddy agents are the agents that are involved in monitoring the PAs. It may be noted that an agent in the model might essay a dual role at the same time. In other words, a PA might be the Buddy for another PA in the model. In previous papers, Protected Agents (PAs) have been referred to as Central Agents (CAs) [10]. As it has been discussed in [10], the schema is currently applicable for only an even number of MAs in the community. Each configuration of
the model is identified by two numerals. The first numeral identifies the total number of PAs in one phase of the cycle. The second number gives the total number of Buddy agents in the phase.

Figure 1 (7-14 Configuration Model)

Figure 1 describes the various interconnections in the first phase of the 7-14 configuration model of the schema. In the 7-14 model there are 14 agents deployed in the field. In each phase 7 agents perform the role of PA and some of them also act as Buddy agents for the other agents within the community. The remaining 7 agents function as exclusive Buddy agents for that particular phase. In the second phase, roles of the Buddy and the PA are reversed. Table 1 documents the agent roles undertaken by the community agents in the 7-14 model. In the table PA denotes Protected Agents and B denotes Buddy Agents. The number attached to PA and B is just to simplify the understanding of the model as it aids in keeping track of the agent in the field. In the schema implementation agents use names with numbers attached to them. For example CommunityAgentOne, CommunityAgentTwo are valid names in the model.

As evident from figure 1 and from the role breakdown described in table 1, all agents perform either of the two roles in some phase of the schema and hence act as protectors and also receive protection. The role assignment and the monitoring of agents in the field are done by the Home Base of the agent community. The implementation model described in the next section uses a single Home Base for the entire agent community. A centralised Home Base for the entire agent community has the advantage of making available a central repository and a reporting centre where information related to the community agents can be easily received and processed.

The Buddy model schema uses a few rules that ensure that there is uniformity in its application across the entire agent community that comes under its purview. In brief these rules can be summarized to state the following. All Protected Agents (PAs) in
the community shall have at least two Buddy agents for itself. The location of the Buddy and the PA shall never be the same at any point of time when the schema is in operation and finally an agent in the community can be a Buddy of only one other agent.

Table 1. Agent role breakdown in the 7-14 Model

<table>
<thead>
<tr>
<th>PA 1st Phase</th>
<th>Buddy 1st Phase</th>
<th>PA 2nd Phase</th>
<th>Buddy 2nd Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>PA1</td>
<td>B6, B7</td>
<td>PA8</td>
<td>B12, B14</td>
</tr>
<tr>
<td>PA2</td>
<td>B1, B5</td>
<td>PA9</td>
<td>B8, B13</td>
</tr>
<tr>
<td>PA3</td>
<td>B2, B4</td>
<td>PA10</td>
<td>B7, B10</td>
</tr>
<tr>
<td>PA4</td>
<td>B8, B9</td>
<td>PA11</td>
<td>B1, B6</td>
</tr>
<tr>
<td>PA5</td>
<td>B3, B10</td>
<td>PA12</td>
<td>B10, B11</td>
</tr>
<tr>
<td>PA6</td>
<td>B11, B12</td>
<td>PA13</td>
<td>B4, B5</td>
</tr>
<tr>
<td>PA7</td>
<td>B13, B14</td>
<td>PA14</td>
<td>B2, B3</td>
</tr>
</tbody>
</table>

Implementation of these rules is carried out by the static agents located at the Home Base of the agent community. After the agent community has been set up and the agent roles have been assigned, the agents are deployed according to pre-decided itineraries. At this point the Buddy model schema is initiated and the first phase is triggered.

In the first phase of the schema, Buddy agents attempt to contact the PA assigned to them. As mentioned before the PA will be located on another agent server, different from the location of the Buddy agent. On making contact, the Buddy agent will attempt to sense the PA. The sense procedure has been described in detail in [10] and hence is not reproduced here. Briefly recapitulating, the Buddy agent in sensing the PA makes contact with it and performs certain checks from the remote location to verify the presence and the integrity of the PA. If the Buddy agent is able to successfully contact the PA and carry out the prescribed checks, it sends a message back to the Home Base with that information. If the Buddy agent is unable to contact the PA, no message is sent back.

Using the reports received at the Home Base, the Confidence Factor (CF) report is generated which gives an indication of the threat perception level in the agent community [10]. The Confidence Factor (CF) function is given as

\[
CF = \left( \frac{n_x + \left( \left( \sum y \right) \ast (-1) \right) \ast (R-r) \right)}{T} \]

Where, 
- \( T \) = Total number of agents within the community
- \( \Sigma x \) = Number of agents operating on Trusted Servers
- \( \Sigma y \) = Number of agents operating on Non-Trusted Servers
- \( R \) = Number of Buddy Reports expected per cycle
- \( r \) = Number of Buddy Reports received per cycle.

In the event that a PA has been killed, the Home Base will receive two messages less than expected and accordingly infer that the PA is missing in the field.
phase is similar to the first phase, except that agent roles are reversed. Buddy agents assume a PA role which implies that apart from sensing other agents, they will also be sensed. It is important to note that the agents to be sensed by the PA are different from those allocated to it in the first phase. On conclusion of the second phase, agents that were involved in the sensing operation, send out a report to the Home Base. The reports are used to generate the CF report as done in the first phase. The CF reporting function is an important part of the schema operation as it allows the Home Base to gauge the current threat perception faced by its agents in the field. Since the agent community is mobile and the location of the agents within the community is a dynamic aspect, it is very necessary to have a reliable tool to monitor the progress of the agents in the field. Since the CF is composed of continuously dynamic parameters, it is capable in reflecting the agent community spread. Another advantage of the CF is that it does not require many parameters for its computation and hence is not overly dependent on any aspect of the community operation. An algorithmic description of the schema has been given in [10].

By employing the Buddy model, the multi-agent community implements a dynamic security layer that alerts the Home Base for possible malicious activity within the community members. However, the schema imposes some stringent demands on the community for it to be effective. The major demand, as imposed by the schema is the requirement of fixed sized agent communities. In other words, at any time for the schema to be effective the community has to adopt a particular configuration, such as the 7-14 model described earlier in the section. Adopting this configuration implies that at any point of time, there should be 14 members within the agent community. While this case is a valid assumption for a fixed size community, it does not consider agents that may wish to terminate themselves on conclusion of their business functionality. In other words, agents that have completed their business functionality are compelled to prolong their existence until other agents of the community have completed their business functions.

To counter this drawback, the paper proposes an extension to the existing model by making a clear demarcation between the business function and the security protection function in the agent community operation. The paper proposes dividing the agent community into two groups. The first group includes all those agents that are involved in carrying out a business task for the agent community. This group shall be referred to the Worker Agent Group (WAG). The second group includes all those agents that implement a security layer for the agent community. This group shall be referred to as the Security Protection Group (SPG). Agents within the SPG do not implement any business function. The sole purpose of their existence within the agent community is to provide a security cover to the WAG. Figure 2 describes the relationship between these two groups within the community.

In figure 2, the agent community boundary is depicted by the dotted outer circle. Agents within the central inner circle that are represented by the dark stars, belong to the SPG. All other agents that do not belong to the SPG circle belong to the WAG. Agents within the WAG are distributed into various clusters. The number of agents within each cluster is dictated by the business functionality of the agent application. In the figure, these agent clusters are shown as circles. Each cluster holds four agents. Since the size of the WAG is variable and agents can leave as well as join the community. To keep the size of agents constant within a cluster there is a regular
distribution of the WAG agents. For example, if 3 agents from one cluster group and 1 agent from another cluster group leave the WAG, the two clusters are merged as one. On the other hand, the size and structure of the SPG remains fixed and is defined by the configuration of the Buddy model adopted. Thus, this extension of the Buddy model defines an additional security relationship.

The first security relationship within the SPG covers the agents within the group. The scope of this relationship is governed by the rules defined for the Buddy model schema operation [10]. The second security relationship in the model bridges the WAG and the SPG and extends the security cover generated by the SPG to the cluster agents of the WAG. Each agent in the SPG is assigned to 1 cluster group of the WAG. Further, each of the assigned SPG agents has a one to one relationship with each of the agents in cluster group. This implies that each SPG agent directly monitors an agent in the assigned WAG group. Since the WAG is a dynamic entity, relationships between the WAG and the SPG are formed and are terminated. This factor is governed by the birth and death of the WAG agents within the agent community. Each agent within the SPG is assigned a set of WAG agents to monitor. The number of agents in this set is variable and is fixed by the Home Base. When any agent belonging to the WAG is terminated, its assigned SPG agent notifies the Home Base. The Home Base can then cross check its agent layout plan and decide if the agent termination was a planned occurrence or the result of a possible malicious attack. If the agent termination was an expected event, it marks the particular cluster for new agent allocation. When a new agent joins the community, the Home Base allocates it to the cluster that is one agent short. The SPG agent monitoring that particular agent cluster is accordingly informed. The next
section gives the schema implementation details and analyses the agent migration of
the SPG agents using performance graphs.

**Buddy Model Schema Operation**

The Buddy model schema is implemented using Grasshopper 2.2.4b mobile agent
systems, running on INTEL P4 machines with 512MB RAM, using Microsoft
used is Sun’s JDK version 14.2_02. Some of the terms used in this section are native
to Grasshopper MAS terminology and their explanation can be found in [3].

**Implementation Details**

The Grasshopper agent system screenshot of figure 3 describes the
SeeTheWorldAgency region which is the central region in the implemented model
of the schema. The scenario described in this section is a news agency scenario. In this
scenario, Agencies register themselves with a particular Region. MAs from different
users and news vendors wishing to purchase or sell news arrive at these registered
agencies and attempt to transact with the agent systems (referred to as Agencies)
directly or with the agents that may be representatives of the agencies.
The 7-14 configuration Buddy model deploys 14 agents over 4 different agencies.
These agents are launched from the Home Base agency of the community. Apart from
the agent identification number agents are identified by their names. In the described
scenario agents are named as CommunityAgentOne, CommunityAgentTwo and so on
unto to CommunityAgentFourteen. In this schema implementation the participating
agents are referred to by their name rather than their identification number just to
make it easier for the readers to trace the agent operation within the schema. It may
be mentioned that the agent identification number is assigned by the agency that
creates the agent and is not an optional parameter while the agent name is assigned a
default value if the agent creator doesn’t assign a name.
The region registering the agencies is identified as SeeTheWorldAgency. Including the
HomeBase-NewsAgency, there are five different agencies that are registered to the
SeeTheWorldAgency region. They are :

- NorthWestNewsAgency
- TransWorldNewsAgency
- TomorrowNeverDiesNewsAgency
- GlobalNewsAgency

The deployed agents are seen docked to their respective agencies in figure 3. Prior to
an attempt to sense, the Buddy agent has to search for the PA at the designated
location. After locating the PA the Buddy agent would sense the agent. After the
sense operation is successfully completed, the Buddy agent contacts the Home Base
reporting a successful contact. This contact-sense phase is repeated after reversing the
roles between the Buddy and the PA.
The Buddy agent’s reports received at the Home Base are used to prepare the
Confidence Factor (CF) report. This function takes into account the operation of all
agents within the community. It uses previous performance and travel histories of agents to differentiate between trusted and non-trusted servers. For example servers which have Service Level Agreements (SLAs) and other business agreements with the Home Base of the agent community are regarded as trusted servers while all other servers are regarded as non-trusted servers. The CF values can be positive as well as negative. A positive CF value indicates a relatively secure level of security for the entire agent community as compared to a negative CF value.

Since the agents are located on different physical machines, it makes it difficult for a malicious action to be carried out on the community agents. Further, since the Buddy agent does not have to move to the same physical location as the PA to sense it, there is no possibility of it being threatened. While some might argue that there is no way to verify the integrity of the results received by the Buddy agent, there remains the fact...
that the PA will have to perform the role of the Buddy agent in the next cycle, from a different location and so if the PA that was sensed has been compromised, it will be unable to participate in the next phase of the schema thereby alerting the sentinels in the Home Base.

Thus changing the location of the agents during the schema operation contributes to the efficiency of the security schema. The next section analyses, the migration factor of the MAs and examines if it has any negative repercussions on the schema operation.

Agent Migration Analysis

The experiments described in this section were undertaken with a view to analyse the effect of the agent migration on the performance of the Buddy model. They attempted to analyse if mobility of agents between phases led to a performance degradation of the Buddy model security schema. Analysis of the mobility context is important as mobility is a central factor in the operation of the Buddy schema. Mobility contributes to the effectiveness and the reliability of the schema since each community agent has an equal role to play in the schema operation. After the execution of its role, the MAs move to a different location and participate in the schema by playing another role. If an agent has been compromised by a malicious server and it is not allowed to migrate, there is an immediate vacuum in the system that is felt by the other agents of the community. In another scenario if the agent is compromised but is allowed to migrate to its next port of call, it will not be able to participate in the schema as it will need to undergo a role play and perform allocated functions. A compromised agent will be unable to perform this operation and again the system will detect a change and the other agent community members will trace the change to its source.

To collect data for the experiments, the agents were supplied with a function that calculated the elapsed time between the time when the Buddy agent starts searching for the assigned PA and the time when the agent is located. The calculated time (in milliseconds) are written into a file, along with the Buddy and the PA details by the Buddy agent. This file is located at the Home Base of the agent community. The data recorded in this file is then used as an input for the Confidence Factor function that indicates the existing level of security within the schema for that particular cycle of the Buddy schema operation.

The graphs of figure 6 describe the performance of the Buddy model under different conditions. The first two graphs in the figure represent the performance of the model when the Buddy agents do not change their locations after the first phase. In other words they carry out both phases of the schema cycle from the same location. The second set of graphs shown as 3 and 4 in the figure, describe the performance of the schema when the Buddy agents change their location after the completion of the first phase. In both the cases, each set of experiments was carried out thrice and are represented by a series each. The first two graphs in the figure show the state in which the Buddy agents did not migrate after contacting the PAs, while in the graphs 3 and 4 they changed their Place location after the completion of the phase. The graphs describe a relatively uniform performance of the schema before and after the implementation of the changed location criteria.
Figure 6. Performance Runs of the Buddy Model
A few variations are observed but since most of them occur during the performance run indicated by series 2, it can be attributed to non-availability of CPU cycles while that particular run was underway. Thus from the agent execution runs described by the graphs in figure 6, it can be inferred that the migration of the Buddy agents has no adverse effect on the performance of the schema. Further this approach of methodically changing the location of the Buddy and PA after each scan would be beneficial in maintaining the integrity of the community agents. The next section enumerates the advantages of the Buddy model by comparing it to other existing agent security schemas.

Related Work

**Cryptographic Security Proposals**

As summarised in [9] security techniques that aimed at protecting the agent from a malicious server used a preventive approach depending on cryptography. This section describes two of the earliest proposals that used such an approach. The first one proposed computing with encrypted functions [17]. This proposal intended to protect the agent from a malicious agent server. The proposal relied on the agent being able to perform computations with encrypted functions at the agent server. These encrypted functions contained an embedded key that allowed the agent to sign the results computed and shift them back to the Home Base, where the results could be decrypted and analysed. Closely following this schema was another schema with a cryptographic base. This schema proposed the use of Obfuscating algorithms and a Time based Black Box approach [16] to protect the agent code. This schema relied on an obfuscating function that would make the agent code indecipherable and discourage the agent server from inspecting the code.

While these two techniques appeared to be strong proposals in securing the agent code from possible tamper attempts, like all other solutions they are dependent on the host platform for execution. This limitation is the biggest contradiction in its success. In the first case, a malicious agent server might reverse engineer the agent code and prevent the security function from triggering at run time. Further the agent server can execute the security mechanism several times in a controlled environment and learn how to invalidate the agent security at run-time in a live environment. Secondly both security proposals rely heavily on an efficient tamper–resistant cryptographic schema to be successful. Such a cryptographic schema would have to be robust enough to prevent hacking by reversible engineering. Thirdly, a cryptographic approach appears to be an expensive proposition for an agent that may require only a few milliseconds to execute its business function. Burdening an agent with a security function that doubles or triples its execution turnaround time and increases dependency on the agent server for resources is a non-viable option and that is where the Buddy model presents a reliable and cost-effective technique for the security of MA communities.

**Non-Cryptographic Security Proposals**
A non-cryptographic approach to agent security usually focuses on detection rather than prevention. While this may appear to be a relatively weaker approach as compared to the previous, it is definitely a cost effective technique as well as a reliable technique of agent security.

Execution tracing [18] was a proposal that required the host agent server to maintain a log of the visiting agent’s execution trace summary, sign the trace with its own signature and using a secure channel transmit it back to the Home Base. While the proposal cannot be called a pure non-cryptographic approach, the core of the proposal did not require any cryptography for its execution. The obvious disadvantage with this solution was its overt dependence on the agent server for its execution and subsequent completion. A malicious agent server will never participate in sending an information log that exposes itself to the Home Base. Secondly, there is no guarantee that the execution log being sent by the host agent server is not a tampered version reflecting the agent’s expected performance. These difficulties make the reliability of the proposal somewhat dubious.

Mutual itinerary recording of agents [12] advocated the use of cooperating agents that recorded the visits of each other. However, this proposal was not robust enough to detect the perpetrator of a malicious action in the event that an agent was terminated. Further, it required two agent servers acting in conjunction could render the schema ineffective. Thirdly, for each agent that was dispatched there was a need for a peer to accompany and to follow its progress and vice versa. This approach contributed to the overheads involved in executing the schema.

The Buddy schema builds over all these documented drawbacks in proposing a solution that involves only the members of the agent community with no overheads involved. The next section documents the advantages Buddy model schema.

**Advantages of the Buddy Model Schema**

1. The schema does not require the host server’s participation in preparing any execution logs or sending any information back to the Home Base. This makes the schema, a comparatively self-reliant solution.
2. The schema involves every member of the agent community in the security function. This approach distributes the workload as well as the risk of the schema being compromised by a hostile action.
3. The schema does not require a cryptographic base making the operation of the schema cost effective.
4. The schema promotes transparency in its operation. This helps in building trust between the visiting agent and the host agent server.
5. The schema presents a reliable approach in enforcing security in agent communities.
6. While the mobility factor in most agent application scenarios is seen to be the cause of increased vulnerability. The schema uses mobility to its advantage by executing the security function from changing locations. This allows the quick detection of a hostile action carried out by a malicious server.
7. The mobility factor of the community agents also makes it possible for the agents to detect the perpetrator of a hostile action while operating in the field. This
detection was not possible in other schemas until the MAs returned to their Home Base and were analysed.

8. The CF report enables the community agents to gauge the security level of the community operation in tangible terms. This is a very useful and important feature in a multi-agent based community operation.

9. The creation of a WAG and SPG groups extends the operation of the Buddy schema to variable sized communities. It allows the creation of variable sized groups within the agent community. The existence of these groups is dictated by business functionality requirements only.

**Conclusion and Future Work**

Security in mobile agent communities is a critical aspect for the success of the agent operation. Unfortunately maintaining a secure environment in a dynamic environment such as those created by MAs requires countering certain limitations and loopholes. The main limitation in proposing a security schema that can protect MAs from malicious host servers is the complete control over the execution environment that the servers hold. This enables malicious servers from inspecting the MA code prior to execution and introducing systems that by-pass the security functions of the MAs.

The Buddy model is a security schema that acts at the agent community level and uses the strength in numbers to counter malicious agent servers. This paper described an extension of the Buddy model schema of security for multi-agent mobile communities. The Buddy model schema is a novel approach to the agent security problem. It focuses on detecting malicious actions against MAs perpetrated by hostile agent servers. The transparency of the schema makes the hosting agent server aware of the security cover. This feature acts as a deterrent and prevents the agent server from launching a malicious attack against the MA.

The paper described an extension of the schema that gives the model flexibility to deal with variable sized communities. It proposed the separation of business functionality from the security function within the agent operation. This approach allows two layers to be formed within the agent community, the Worker Agent Group (WAG) and the Security Protection Group (SPG). The operation objective of both these groups is independent of each other. However, these groups co-exist in terms of a one-to-many relationship that is exerted by the SPG agents on the WAG agents. The new approach makes the Buddy model schema operation relatively flexible and enables broader security coverage within the agent community.

Further, the paper argued that the Buddy model schema was not tied down by the migratory nature of the community agents. To prove this, a set of experiments were carried out on the agent community. In the first set of experiments the location parameter remained unchanged in both phases, while in the second set of experiments; the location of the agents was changed between the two phases. The performance results obtained from the two set of experiments demonstrated a stable and a relatively unchanged performance of the schema operation when migration of agents was introduced. Thus, the mobility factor in the Buddy model schema does not affect the operation of the MAs. On the contrary as described in the
paper, mobility allows the detection of a malicious action on a MA while they are still operating in the field. The paper also compared the Buddy model schema against some existing security proposals and enumerated its advantages. Future work will focus on extracting other context parameters that influence the operation of the Buddy model between the two groups and studying their effects on the schema.

References


