Mobile Agent Security using Reference Monitor-based Security Framework

Sandhya Armoogum, Nawaz Mohamudally
Dept. Industrial Systems & Engineering
University of Technology, Mauritius (UTM)
La Tour Koenig, Mauritius
email: asandya@umail.utm.ac.mu

Nimal Nissanke
Emeritus Professor,
London South Bank University, London, UK
email: nissanke@gmail.com

Abstract— **In distributed systems and in open systems such as the Internet, often mobile code has to run on unknown and potentially hostile hosts. Mobile code, such as a mobile agent is vulnerable when executing on remote hosts. The mobile agent may be subjected to various attacks such as tampering, inspection, and replay attack by a malicious host. Much research has been done to provide solutions for various security problems, such as authentication of mobile agent and hosts, integrity and confidentiality of the data carried by the mobile agent. Many of such proposed solutions in literature are not suitable for open systems whereby the mobile code arrives and executes on a host which is not known and trusted by the mobile agent owner. In this paper, we propose the adoption of the reference monitor by hosts in an open system for providing trust and security for mobile code execution. A secure protocol for the distribution of the reference monitor entity is described as well as a novel approach to assess the authenticity and integrity of the reference monitor running on the destination agent platform before any mobile agent migrates to that destination. This reference monitor entity on the remote host may provide several security services such as authentication, integrity and confidentiality of the agent's code and/or data.**

Keywords- Security; Mobile agents; Reference monitor, Trust

I. INTRODUCTION

During the last decade, there have been major changes in distributed computing. Programs are no longer constrained to execute on one machine. Code can now be migrated to other hosts for execution. The most well-known example of mobile code is the use of Java applets and JavaScript code in a Web browser. This form of code mobility is referred to as code on demand. Code mobility has many uses, i.e., vendors can use mobile code to reconfigure software, Microsoft uses mobile code to distribute software patches, mobile code can also be used to manage distributed system by performing load balancing [1]. Mobile agents represent a more sophisticated and powerful form of code mobility. A mobile agent is a program that can move from one host to another according to its own internal logic. Such mobile agents can have weak or strong mobility.

Mobile agent computing paradigm presents numerous advantages, compared to the traditional client-server based computing model, which include reduced network usage, better fault tolerance, adaptability to changes in the environment, and platform independence [2]. Thus, the mobile agent computing paradigm has become a natural and flexible way for implementing many applications on the network such as e-commerce and auctioning, network monitoring, real-time control systems and cloud computing. Recently, mobile agent computing proves to be very useful in the context of wireless sensor network (WSN) [3]. As such, the mobile agent computing paradigm provides a very intuitive and flexible approach to solving old and new problems arising in many areas of computing (e.g. intrusion detection [4], web crawling [5], Big Data analysis [6], searching and communications in Internet of Things [7] and e-learning [8] due to their mobility and autonomous nature, their ability to learn and adapt to changing environments, their ability to communicate and collaborate with one another to perform some complex task, and their ability to clone themselves, if needed.

However, for mobile agent applications to be widely adopted, some security issues have to be addressed, as mobile agent applications, mainly deployed in open environments, may possibly be exposed to attacks. Currently, the lack of an integrated security solution is a main drawback, which has to be tackled before the mobile agent computing paradigm is widely accepted by industry. Four threat categories have been identified in [9] as follows: (1) mobile agent attacking an agent platform (AP), (2) AP attacking a mobile agent, (3) a mobile agent attacking another agent on the AP, and (4) other entities attacking the agent system. Some practical solutions exist for securing AP against malicious mobile agents. However, securing the mobile agents against malicious AP is more challenging as the host has full control of the execution environment. Moreover, the mobile agent computing model violates some of the fundamental assumptions of conventional security techniques, i.e., programs runs on hosts which are trusted. Other important assumptions are the identity and intention assumptions [10]. Usually, when a program attempts some action, the program acts on behalf of a known user and it is assumed that the person intends the action to be taken. When mobile agents execute on unknown hosts, then neither identity or intention of the host is clear.

The fact that mobile agents may have to operate in such unknown and open environment requires the concept of trust of the hosting agent platforms despite that they may be previously unknown to the mobile agent's owner [11]. In this paper, we propose an approach for building trust in a mobile agent system by exploiting the reference monitor concept which provides trust of unknown APs and subsequently trust enhanced security. Such a reference monitor entity is obtained from a trusted third party (TTP) and can provide or arbitrate several security services resulting in a system which supports mobile agent applications in an open system.

The rest of this paper is structured as follows. Section II discusses related work. Section III presents the proposed security framework based on trust provided by the Reference Monitor for providing a trusted computing environment. Section IV discusses the integration and evaluation of the reference monitor entity in an agent platform. Section V presents the experimental setup used and results. Finally, Section VI presents the conclusions drawn and outlines future work.

II. RELATED WORK

Several security mechanisms exist to secure mobile agents as they execute on remote hosts. However, most of the existing schemes mainly allows the detection of integrity attacks on the mobile agent and none of these solutions provide a comprehensive integrated security framework for protecting mobile agents.

One approach for mobile agents to have comprehensive security against malicious platform involves allowing mobile agents to migrate to known and trusted APs only [12]. Then, agents are sent in encrypted form from one trusted AP to another, where they execute, often after authentication of the AP. However, such an approach seriously limits the agent’s execution on a limited number of known and trusted APs. It is also not suitable for some applications like search agents (searching for information) on the internet, and mobile e-commerce agents where AP may not always be known and is against the notion of an open multi-agent system where new APs can be dynamically added or removed from an agent's itinerary.

Another proposed solution for providing comprehensive security to mobile code advocates the use of trusted, tamper-proof hardware which is not controlled by the local system and which supports secure mobile agents execution [13]. This secure trusted computing base on each AP, thus, provides the trusted environment for running critical code of the mobile agent. Here, mobile agents move from one trusted environment to another. Local resources on the system are accessed in a client-server mode. The system outside the trusted hardware has no access and thus cannot interfere with the execution of the mobile agent. The external system can only interact with the trusted tamper resistant hardware via some restricted interface. The main drawback of this approach is that every host has to be supplied with secure trusted hardware which is not a simple task as such hardware installation and maintenance may be expensive. Furthermore, the use of tamper-proof hardware may not scale-up efficiently and may be limited to highly security-sensitive mobile agent computing areas such as banks and stock markets. Moreover, the tamper-resistant devices could also become a performance bottleneck in the execution of mobile agent especially in cases where smart cards are used as a cheap alternative for providing hardware trusted computing base [14].

Another approach for protecting mobile agents involves code obfuscation, whereby the agent's program is made illegible and data hidden, thus rendering it difficult to read and modify the agent code, data and partial results [15]. An obfuscated mobile agent is like a black-box entity; it only permits AP to provide inputs and read outputs from the mobile agent. Thus, even if the mobile code runs on unknown and untrusted APs, it can maintain confidentiality and integrity. However, this technique often only provides limited code confidentiality as given time, the malicious AP will be able to de-obfuscate the code. In [16], the authors actually show that complete obfuscation is impossible. Furthermore, the malicious host could still re-execute (replay) the mobile agent several times so as to observe its reaction and guess its decision making strategy for example. Esparza, et al. [17] proposes to monitor the execution time on an AP. A longer than expected execution time on an AP is indication that the AP may be attempting to de-obfuscate the agent, modify the agent code, data and/or partial results or replay the mobile agent. The main drawbacks of the approach are that it requires the agent to return with the partial results from each host visited to the Home Agent Platform (HAP), i.e., HAP must be connected until the transaction is over. The application would then no longer support disconnected and asynchronous processing as promised by mobile agent technology, though it does provide security of mobile agent such as integrity, execution integrity, and detection of Denial of service (DoS) attack.

Finally, the use of mobile cryptography (also referred to as function hiding) which aims to offer provably strong protection to mobile agents against both modification and inspection attacks has been proposed. Mobile cryptography is such that a mobile agent program is encrypted into a ciphered executable program where it can execute on the untrusted AP while remaining in the ciphered form [18] [19]. The efficiency of this approach is unknown and to the best of our knowledge, there is no practical implementation. It is still unclear if such a scheme can be implemented for real-life applications. It is thus obvious that securing mobile agent remains a challenging research problem.

Trust is an important component of mobile code security [20]. When agents are executing in open environments on unknown and untrusted APs, if they could have some guarantees of trust, this would provide a basis for better security solutions. In [21], the authors describe a trust management architecture (MobileTrust) that can be developed to manage security related trust relationships explicitly and to make trust decisions. In such a system, trust management brings an improvement in security as by leveraging the trust knowledge gained on the past behaviours of other execution hosts, the mobile agents itinerary can be composed such as to minimize security attacks from potential malicious APs. Similar trust computation can allow the AP to evaluate the trustworthiness of a mobile agent from a specific agent owner. However, such a technique may not be suitable for all open environment applications. For example, in a mobile agent based e-commerce application, a mobile agent from an owner may not visit the same AP twice and thus the trust values of the current execution may never be useful. In [22], a trusted third party called a Clearing House is proposed to maintain trust information about APs. Before a mobile agent migrates to an AP, trust level associated with the AP may be found out. However, this solution heavily relies on the Clearing House which keep tracks of the trustworthiness of each AP and also may not be suitable for applications where the mobile agent have a dynamic itinerary. Finally, such mechanisms of monitoring behaviour of AP may be unpractical since not all attacks can be detected and thus reported, especially breach in agent data and code confidentiality.

III. Reference Monitor Based Security Framework

In this research work, we propose a practical and pragmatic technique for providing mobile agent security based on the reference monitor (RM) concept. Since its introduction, in the early 1970s, in the "Anderson Report" [23], the RM concept has been adopted for securing computer and network. This concept visualises a system component, called a *reference validation mechanism*, to be responsible for administering the system's security policy. It thus defines the requirements for implementing such a mechanism in a manner that ensures that malicious entities cannot circumvent policy enforcement [24]. The RM concept thus provides a trusted and verifiable security policy enforcement mechanism[25]. This is in line with the U.S Government's criteria for building secure systems, the Trusted Computer System Evaluation Criteria (the Orange book) [26], where the reference monitor is mentioned. The diagram in Figure 1 depicts the logical structure of the RM.

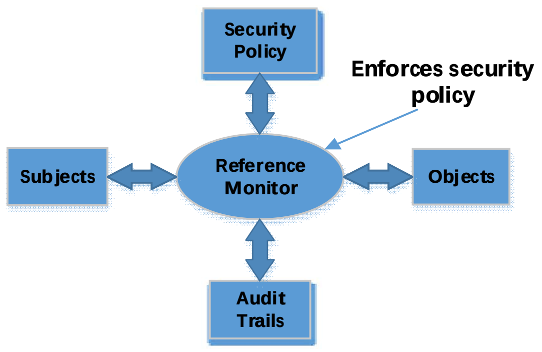

Figure 1. Logical Structure of RM

Implementations of the reference monitor concept make use of several traditional security measures, which apply to the agent environment. Such conventional techniques include cryptographic methods for authentication of AP/agent, encryption of data, integrity of data, and access control methods, thereby providing an integrated security solution. Thus, an implementation of the RM can be used to enforce security. The security policy for agent computing can involve different security requirements, however, any AP which implements an RM entity can be considered to be trustworthy. The mobile agent may then migrate to such an AP for secure execution. Thus, the presence of an RM entity which enforces security suffices to turn any remote and unknown host into a trusted computing base in an open system. In the case of mobile agent computing, we propose that the RM entity be in the form of an agent (RM-agent) which runs on the each AP. We define the following fundamental assumptions about the trust association of all entities in the security framework, for providing mobile agent security. These trust associations comprise the trust model in our system.

- The Certificate Authority (CA) authenticates principals and issues certificates to entities such as the agent owner, AP administrator, RM-agent, trusted-third party (TTP) RM entity distributor. Users/agent owners and APs with valid certificates are considered trustworthy.
- Trusted Third Parties (TTPs) are principals with genuine certificates from CA and are trustworthy. These TTPs distribute standard RM-agents, which satisfy some design specifications, for use by any AP. Such TTPs can also act as subordinate CAs. The trusted CA delegates them the right to issue certificates to APs. Then, the TTP can also maintain a directory server where it stores information about each AP and their corresponding RM-agent.
- RM-agents on APs with genuine certificates from the TTP are trustworthy. These certificates are signed by the TTP who must have full knowledge of the RM agent's behaviour and capabilities and thus all the possible consequences of its operation. The system administrator of the AP can request for an RM-agent and its certificate from the TTP,
- Agents signed by trustworthy owners are trustworthy. The agent-owner is expected to have knowledge of the capability and behaviour of the agent and to take responsibility of the actions of the agent acting on behalf of the user/owner.

The underlying idea of our trust model is based on the usage of undeniable proofs like digital signatures, i.e., RM-agents are signed by the TTP (using the private key of the TTP), thus ensuring that the RM-agents are genuinely from the TTP. The digital signature of the RM-agent also allows to check the integrity of the RM-agent, i.e., indication of tampering, if any, on the RM-agent. In the next section, we describe how the RM-agent can be delivered securely to an AP and how a mobile agent can assess the authenticity and integrity of the RM-agent before deciding to migrate to the remote AP. In this work, we focus on the RM integration into the AP to establish trust, rather than on how the RM-agent enforces the security policy to provide security to mobile agents executing on the AP.

IV. Integration and Evaluation of RM-Agent

In this section, we describe how the RM-Agent, which embodies the reference monitor concepts, can be securely obtained and integrated in an AP to establish trust so as a mobile agent can safely migrate and execute on the destination AP.

A. Distribution and Integration of RM-agent into an AP

The different steps required for an AP to obtain a trusted RM-agent from a TTP in a secure manner is illustrated in Figure 2.

Step 1: AP registers with TTP and requests an RM-agent for enforcing security policy. The security to be provided by the RM-agent, and thus the AP, may vary based on the security policy of the AP.

Step 2: TTP generates a digital certificate ($RM_{TTPCert}$) for the RM-agent. This certificate contains information that would allow the receiving AP to verify the authenticity and integrity of the received RM-agent from TTP. Some important information on the $RM_{TTPCert}$ certificate includes: identity of AP to whom RM-agent is distributed; cryptographic hash value of RM-agent code; digital signature of the RM-agent, version no. of RM-agent, and security policy ID implemented by RM-agent. The $RM_{TTPCert}$ certificate is itself signed by the TTP.

Step 3: The TTP sends the RM-agent and the $RM_{TTPCert}$ certificate to the AP administrator/owner.

Step 4: Upon receipt of the RM-agent code and $RM_{TTPCert}$ certificate from the TTP, the AP administrator/owner can verify that the signature of the certificate is correct. The AP administrator/owner then uses the digital signature of the RM-agent, provided in the $RM_{TTPCert}$ to ensure that RM-agent was sent by the TTP (authenticity check). Finally, the AP administrator/owner, checks if the hash of the RM-agent file downloaded is the same as that on the certificate to ensure that the RM-agent has not been replaced or tampered with (integrity check). AP administrator/owner also checks other information on the certificate.

Step 5: Finally, the AP administrator integrates the RM-agent in the AP to establish the AP as a trusted entity which provides security as per the security policy ID.

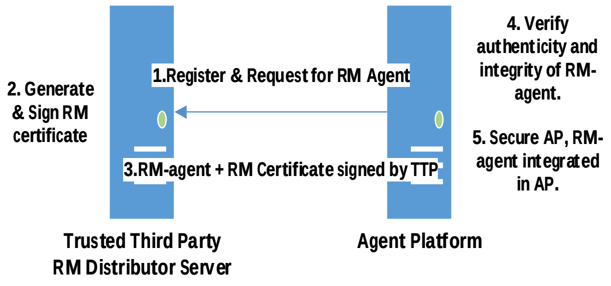
2. Generate & Sign RM certificate
1.Register & Request for RM Agent
3.RM-agent + RM Certificate signed by TTP
4. Verify authenticity and integrity of RM-agent.
5. Secure AP, RM-agent integrated in AP.
Trusted Third Party RM Distributor Server
Agent Platform

Figure 2. Distribution of RM-agent by TTP distributor

Thus, it can be seen that the RM-agent can be easily and securely downloaded from a TTP by any AP in an open system and integrated into the agent execution environment to provide trust enhanced security.

B. *Verifying Authenticity and Integrity of RM-agent*

Before a mobile agent migrates to the destination AP, if security is required, it has to ensure that RM-agent running on the destination AP has not been modified and is truly from a TTP. This problem is heightened by the fact that in an open system, the destination AP may not be known by the mobile agent. Verifying the trustworthiness of the RM-agent on a remote AP is necessary as one of the property of the RM is that it should be tamperproof and verifiable. It is true that the RM-agent has a certificate, provided by the source TTP ($RM_{TTPCert}$), but the certificate is no guarantee that the RM-agent running on the AP has not been modified. The $RM_{TTPCert}$ from the destination RM-agent allows the mobile agent to check the signature of the certificate to determine if the certificate and the data it contains are genuine and unmodified. The certificate shows that the RMagent was obtained from a TTP. The certificate also contains the security policy ID which informs the agent of the security provided by the destination AP. However, neither the $RM_{TTPCert}$ certificate nor the digital signature of the RM-agent allows to verify that the RM-agent *running* on the destination AP is the same as the one distributed by the TTP. Thus, it is not sufficient to confirm the trustworthiness of the RM-agent on the destination AP.

Alternatively, if the mobile agent could compare the hash value of the RM-agent code from the certificate with the hash code of the actual RM-agent to verify authenticity and integrity of the RM entity. However, given that the RM-agent is on the destination AP, the mobile agent on a source AP cannot calculate the hash value of the RM-agent. The mobile agent cannot also rely on the hash value received by a destination AP, as the destination AP may send a stored hash value of RM-agent, while it is running a modified version of the RM-agent. In such circumstances, the static validation approach fails to allow the mobile agent on a source AP to determine the integrity and authenticity of the RM-agent running on the destination AP prior to migration. Verifying the integrity and authenticity of code running at a remote destination is a challenging issue which we address as follows.

We identify the following requirements regarding the verification of the integrity and authenticity of the RM-agent running on the destination AP:

1. The source AP should be able to verify the integrity and authenticity of the RM-agent on the destination AP by storing minimal information about the RM-agent.
2. The integrity and authenticity checking mechanism has to stay secure even if the destination AP is malicious.
3. The communication bandwidth used during the verification process should be minimal, i.e., it should not involve the transfer of large amount of data.
4. The verification mechanism should be efficient in terms of computation.
5. It should be possible to run the verification several times, if desired, to ensure that integrity and authenticity of the RM-agent is maintained at all times. Static validation actually fails to satisfy this requirement because the hash code can be stored and sent back by the destination AP whenever required.

We propose a dynamic validation mechanism that satisfies the above conditions based on the challenge response approach. The mobile agent on a source AP can thus effectively verify the integrity and authenticity of the RM-agent on the destination AP that it wishes to migrate to.

C. *Authentication and Integrity Checking Protocol (AICP)*

The task of checking the integrity and authenticity of the RM-agent on the destination AP in our security framework lies upon the RM-agent on the source AP, since it is a service that may be requested by any mobile agent. If the mobile agents themselves were to perform this task, it would have added unnecessary burden on the mobile agent. The mobile agents thus remain lightweight and are programmed to only perform their tasks in the application. Accordingly, when a mobile agent wishes to migrate from its current AP to another, it requests the RM-agent on the source AP to assess the security of the destination AP. Given that mobile agents can only interact with one another by communicating using an Agent Communication Language (i.e. an agent cannot invoke a method of another agent but it can request the destination agent for some processing), the proposed dynamic validation of authenticity and integrity of the RM-agent involves interaction which we refer to as the *Authentication and Integrity Checking protocol (AICP).*

This interaction begins with the source RM-agent requesting the $RM_{TTPCert}$ certificate from the destination RM-agent. From the certificate, the source RM-agent learns about the TTP from which the destination RM-agent has been acquired. Based on this knowledge, the source RM-agent can retrieve the distributed RM-agent files *from the TTP* (same files/codes are running on the destination AP as the destination RM-agent), so that the hash value of the RM-agent can be computed locally. It is assumed that the TTP makes available the RM-agent file for download for such verification process. It may be argued that downloading the RM-agent code for this purpose may require a large amount of bandwidth and storage space on the source AP. Nonetheless, our implementation of the RM-agent in JADE results in an RM-agent Java class file which is 17 KB in size. JADE is used as it is one of the most popular APs and it is Java-based; most of the existing APs are Java-based APs. Given the increasing bandwidth capacity and high speed of today's network, retrieving and caching the RM-agent for a short period of time should not pose any problem. Assuming there are ten different TTP sources with different implementations of the RM-agent, and that each RM-agent is of the size 20KB, for an AP to temporarily cache all the RM-agents, only 200KB (0.2 MB) of storage space is required. Thus the proposed protocol is storage efficient.

Next, the source RM-agent issues a challenge (random number -R) to the destination RM-agent. This challenge (R) can be sent encrypted using the public key of the destination RM-agent. The source RM-agent also calculates the expected response as follows and as depicted by Figure 3.

- Challenge (R) is concatenated with the RM-agent class file (RMCF): R || RMCF
- The concatenated output is hashed to obtain the hash value H(R || RCMF)

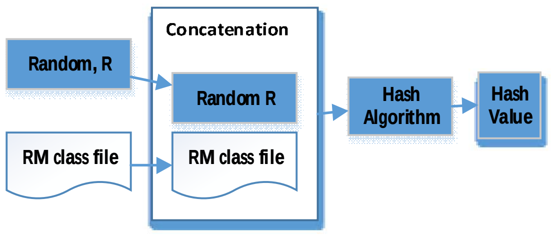

Figure 3. Calculating the response of the challenge at source

Similarly, the destination RM-agent uses its Private key to decrypt the encrypted Challenge sent by the source RM-agent. This step ensures that only the destination RM-agent has access to the Challenge (R). The destination RM-agent concatenates the Random number R to the current actual RM-agent file and then calculates the hash value of the concatenated input as shown in Figure 4. The response from the destination RM-agent is compared with the expected response computed by the source RM-agent. If they matches, then it is safe to assume that the destination RM-agent has not been tampered with (integrity). If the two hash values are the same, we can also assume authenticity of the destination RM-agent as the hash value has been computed using the RM-agent file from the TTP.

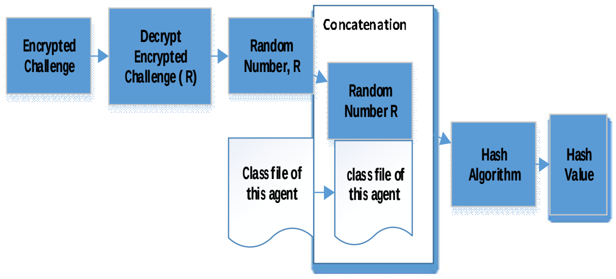

Figure 4. Calculating the response of the challenge at destination

V. Experiments and Results

Experiments were designed with JADE agent development environment. The destination RM-agent uses its **own class file** for generating the Hash value. Using the this.getClass() method allows to find the agent's Java class filename. Once the class filename is known, the path of the file can be found. Because the RM-agent is running, the RM-agent java class file is read-only, i.e., cannot be modified. We read the file and copy it into another file, e.g., CopyFile. Concatenation is implemented by appending the Random number to the "CopyFile". The "CopyFile" is then hashed using SHA-256 to generate the hash value. Thus, it can be observed that the hash value generated truly correspond to the destination RM-agent. Similarly, on the source RM-agent, the RM-agent class file is read and copied into a new file, to which the random number is appended in the end (concatenation). This new file is passed as an argument to the hash function to generate the expected hash value. The same hash function is used by the source RM-agent and the destination RM-agent.

Given that in the context of agent computing all interaction takes place by means of ACL messaging between agents, the AICP is implemented as a series of messages exchanged between the source RM-agent and the destination RM-agent. The RM-agent is programmed by adding the following two behaviours: *InitiatorAICP* and *ResponseAICP* implemented as a OneShotBehaviour.

It was assumed that both the source AP and destination AP uses the same TTP source for the RM-agent. We run the AICP, with different versions of the destination RM-agent class file. It was observed that when the proper destination RM-agent was used at the destination AP, the response obtained was as expected. However, when the destination RM-agent code has been changed, the response obtained was different than the expected response. Thus, using the AICP, a source AP may successfully assess the authenticity and integrity of the destination RM-agent and consequently the trustworthiness of the destination AP.

VI. Conclusions and Future Work

We introduced the RM-agent, as a fundamental component of our framework. The RM-agent allows to implement the reference monitor concept, whereby a reference validation mechanism (implemented by the RM-agent), is responsible for enforcing the system's security policy. The RM-agent satisfies the requirements of the reference monitor concept in the sense that the mobile agent can verify that the RM-agent is from a trusted source (RM-agent is verifiable and trustworthy) and it is hasn't been tampered with. A novel dynamic verification approach was proposed for assessing authenticity and integrity of the remote RM-agent running on the destination AP.

RM-agents contribute to providing a secure computing environment to mobile agents. Altogether, with the security framework, it is possible for a mobile agent to migrate to any AP with the assurance of security. In this case, it is not important for a mobile agent to know every AP, as long as the AP deploys a RM-agent obtained from a TTP. Thus, our security framework is most suitable for open systems. Additionally, the security framework supports mobile agent computing with static as well as dynamic itineraries. Future works, revolves around equipping the RM-agent to ensure different security services by integrating different behaviours to the RM-agent. Then, the RM-agent will be able to enforce security such as code and data confidentiality, integrity, execution integrity.

References

[1] R. R. Brooks, "Mobile Code Paradigms and Security Issues", IEEE Internet Computing, vol.8, no. 3, pp. 54-59, May/June 2004

[2] D. B. Lange and M. Oshima. "Seven good reasons for mobile agents." Communications of ACM , March 1999, 88-89.

[3] R. K Verma and S. Jangra, S., "Significance of Mobile Agent in Wireless Sensor Network". International Journal of Advance Research in Computer Science and Management Studies, 1(7), 2013, pp. 328-335.

[4] T.T. Khose Patil, and C, Banchhor, "Distributed Intrusion Detection System using mobile agent in LAN Environment." International Journal of Advanced Research in Computer and Communication Engineering, 2(4), 2013. pp. 1901-1903.

[5] V. Upadhyay, J. Balwan, G. Shankar, and Amritpal, "A Security Approach for Mobile Agent Based Crawler." Proceedings of the Second International Conference on Computer Science, Engineering & Applications (ICCSEA 2012) New Delhi, India, 2012.

[6] Y. Essa, G. Attiya, and A. El-Sayed, "Mobile Agent Based New Framework for Improving Big Data Analysis." IEEE International Conference on Cloud Computing and Big Data (CloudCom-Asia). Fuzhou, 2013.

[7] W. Godfrey, S. Jha, and S. Nair, "On a Mobile Agent Framework for an Internet of Things". IEEE International Conference on Communication Systems and Network Technologies (CSNT), 2013

[8] M. Higashino et al., "Management of Streaming Multimedia Content using Mobile Agent Technology on pure P2P-based Distributed e-Learning System. Barcelona", In the Proceedings of the 27th IEEE International Conference on Advanced Information Networking and Applications, 2013

[9] W. Jansen, and T, Karygiannis, " NIST Special Publication 800-19-Mobile Agent Security", Technical paper, Computer Security Division: National Institute of Standards and Technology, 2000.

[10] C. Lin, V. Varadharajan, "Trust Enhanced Security - A New Philosophy for Secure Collaboration of Mobile Agents", COLCOM, 2006, International Conference on Collaborative Computing: Networking, Applications and Worksharing, International Conference on Collaborative Computing: Networking, Applications and Worksharing 2006, pp. 76.

[11] Jensen, C. Damsgaard. "The Importance of Trust in Computer Security." Trust Management VIII. Springer Berlin Heidelberg, 2014. pp 1-12.

[12] X. Guan, Y. Yang, and J. You, "POM-A mobile agent security model against malicious hosts". IEEE Fourth International Conference on High Performance Computing in Asia-Pacific Region. Beijing, China, 2000

[13] S. Zhidong, and T. Qiang, "A Security Technology for Mobile Agent System Improved by Trusted Computing Platform." Proceedings of the Ninth International Conference on Hybrid Intelligent Systems (HIS 2009) , Shenyang, 2009.

[14] S. Loureiro, and R. Molva, "Mobile code protection with smartcards." Proceedings of the Sixth ECOOP Workshop on Mobile Object Systems: Operating System Support, Security and Programming Languages, Cannes, France,, 2000.

[15] S. Armoogum, and A. Caully, "Obfuscation Techniques for Mobile Agent code confidentiality." Journal of Information & Systems Management, 1(1), 2011 pp. 25-36.

[16] B. Barak, et al. "On the (im)possibility of obfuscating programs." Proceedings of the 21st Annual International Cryptology Conference - Advances in Cryptology, Santa Barbara, California, USA, 2001, pp. pp 1-18.

[17] O. Esparza, M. Soriano, J. L. Munoz, and J. Forne, "A protocol for detecting malicious hosts based on limiting the execution time of mobile agents." IEEE Eight International Symposium on Computers and Communication (ISCC'03), Kemer-Antalya, Turkey, 2003

[18] T. Sander, and C. F. Tschudin, "Towards Mobile Cryptography." Proceedings of IEEE Symposium on Security and Privacy. Oakland, CA, 1998

[19] H. Lee, J. Alves-Foss, and S. Harrison, "The use of Encrypted Functions for Mobile Agent Security." IEE 37th International Conference on System Sciences, Hawaii, 2004

[20] U. G. Wilhelm, S. Staamann, and L. Buttyn. "On the problem of trust in mobile agent systems." In Proceedings of Network and Distributed Security Symposium, San Diego, California, Internet Society, 1998

[21] C. Lin, V. Varadharajan, "MobileTrust: a trust enhanced security architecture for mobile agent systems", International Journal of Information Security, Volume 9, Issue 3, 2010, pp 153-178

[22] D. Foster, V. Varadharajan, " Security and trust enhanced mobile agent based system design," Third International Conference on Information Technology and Applications. 2005, Vol 1 pp 155 - 160

[23] J. Anderson, " Computer Security Technology Planning Study, Bedford MA: Technical Report ESD-TR-73-51", Air Force Electronic Systems Division, Hanscom AFB. 1972

[24] Jaeger, T., 2011. Reference Monitor. Encyclopedia of Cryptography and Security (2nd Ed.), 2(1), pp. 1038-1040.

[25] C. E. Irvine, "The Reference Monitor Concept as a Unifying Principle in Computer Security Education." Proceedings of the International Federation for Information Processing (IFIP 99) & 1st World Conference on Information Security Education. Kista, Sweden, 1999

[26] D. o. D., "Trusted Computer System Evaluation Criteria, Orange Book", Library No. S225 711: Dept of Defense CSC-STD-001-83. 1983