SOLUTIONS TO MOBILE AGENT SECURITY ISSUES IN OPEN MULTI-AGENT SYSTEMS

Olumide Simeon Ogunnusi, Olasunkanmi Okunola Ogunlola
Department of Computer Studies, The Federal Polytechnic,Ado-Ekiti, Ekiti State, Nigeria
olu_simmy@yahoo.com, okunola_olaq@yahoo.com

Abstract- *Research has unfolded that mobile agents are veritable tool to combat challenges posed by distributed systems. This is due to the unique attributes of mobile agents such as mobility, latency reduction, autonomy and ability to transport process to remote data repository and take results back to their principals. Despite all these appealing benefits of mobile agent, several security issues are associated to it which needs to be addressed to trigger its widespread application. This has made this aspect of distributed computing a hot research area especially in the academic scene. This article presents a review of the security issues associated to mobile agent paradigm in open multi-agent systems. It involves threats from agent to agent, agent to agent platform, agent platform to agent, and others to agent platform. The article also discusses the threats' requirement and their respective security solutions.*

Keywords: *mobile agent; agent platform; mobile agent security; open multi-agent system*,

1 Introduction

An agent that has the ability to move from one host to another on a network (logical mobility/code mobility) is termed a mobile agent [1].It is an autonomous software that hop through a computer network to perform some computation or collect information on behalf of a human user/application [2]. Over a decade, research activities have revealed that mobile agent is a veritable tool to overcome the challenges posed by distributed systems as a result of its autonomy, latency reduction and capability to move from one host to another, encapsulating its codes, data and/or state. It engages in strong mobility by encapsulating its code, data and state and migrates from one host to another in a computer network. Weak mobility of mobile agent involves only the migration of its code and data.Mobile agents are found to have relevance in some specific applications such as electronic commerce (stock markets and electronic auctions) [3-5], information retrieval and dissemination [6-8], personal assistance [9-11].

An open multi-agent system (MAS) is that in which mobile agents owned by various principals can enter and leave the system at any time [12]. In an open multi agent system, mobile agents traverse network freely to access resources and services required to attain their goals. The free hopping of mobile agents in open multi-agent systems makes trust and access control inevitable to guarantee the security of mobile agents, agent platforms and agent communication. Not until these are fully and reliably achieved, widespread acceptance of mobile agent application in commercial distributed systems might be a mirage. The runtime environment where an agent originates is called the home platform for the agent and is considered as trusted. Every other platform in the mobile agent's itinerary is untrusted and hence an efficient trust mechanism is needed to establish the trust level of such platform for the safety of the agent.

Researchers have proposed several techniques to guarantee secured transactions between two entities such as between two mobile agents and between mobile agent and an agent platform. These techniques are discussed in Section 5. Generally, some basic ingredients are fundamental to facilitate the migration and entities interactions across an open network such as Internet. These are[13]: (1) Common agent execution language (2) Process persistence (3) Communication mechanism between hosts and (4) Protection of agents and host. Other important ones are: (1) Common agent communication language (2) Protection of agent communication.The runtime environment of mobile agent takes full control of the agent during its execution and as such a malicious host can take advantage of this to attack the agent and compromise the confidential information carried by the mobile agent.

The rest of the paper is organized as follows: Section 2 discusses the key security properties of mobile agents while Section 3 presents the requirements for mobile agent security. Security threats to mobile agents in open multi-agent systems are discussed in Section 4. Detail discussion of the solutions to mobile agent threats is presented in Section 5. These solutions are classified into *detection* and *prevention*in Section 6 while Section 7 concludes the paper.

2 KEY SECURITY PROPERTIES OF MOBILE AGENTS

The protection of mobile agent-based systems needs to guarantee the following five key security properties [14]:*Confidentiality, Integrity, Accountability, Availability*, and*Non-repudiation.*

2.1 Confidentiality

Confidentiality ensures that the private data of a mobile agent stored in the agent program or on its platform must be personal to the agent. That is, no unauthorized person can read or manipulate it without permission. The mobile agent framework ensures that their local and remote communication is kept private. An external entity (snooper) can create fake information for its own profit. For example, the goal of an agent might be to collect information on air ticket and to pursue this mission, the agent has to visit different airline service providers and collect various information such as departure time of airbus and flight charges. Any of the airlines visited by the agent can create a malicious agent to change the information carried by the mobile agent such as altering the flight charges of the cheapest airline for its own benefit.

A number of security schemes have been proposed for the confidentiality protection of mobile agent such as code obfuscation [15], environmental key generation [16], Access control [17], and homomorphic encryption [18].

2.2 Integrity

Integrity is an agent security requirement which ensures that the agent code, data, and state must not be modified by an unauthorized agent or platform. That is, only an authorized entity should be privileged to modify the agent contents. Mobile agent itself cannot restrain malicious agent or platform from changing its state, code or data but can only detect the alteration ([19]. In an agent system, there is need for some access control policies to restrict unauthorized users to curtail goal-oriented attacks such as deletion of unfavourable information, reuse of old information, change of source and destination addresses and the likes. Researchers have proposed many schemes for the protection of mobile agent components. For example Farmer, Guttman [20] proposed an architectural model for trust relation between the principals of mobile agent systems. In this model, state appraisal formed a unique aspect of the architecture that shields users and hosts from attacks via agent modifications. Cao and Lu [21] proposed an access control model which captures the path history of the agent and arrived at a decision according to whether the captured path matches a path pattern set and whether the path can follow a host patch set. Mitrovic and Arribalzaga [22] proposed an architecture for securing mobile agent systems using trusted domain and proxy agents. This concept uses proxy agents to enable transparent and secured services to both the security-aware agents and legacy agents.

2.3 Accountability

In mobile agent system, accountability is concern with ensuring that all the entities of mobile agent system comprising user, process host and agent itself must be accountable for their activity and they must be uniquely identified, authenticated and audited. To achieve this, the agent system must ensure that every security related event performed by an entity is stored in the platform security policy in a defined information format like agent name, time of event, security event, success/failure status of the event.

2.4 Availability

Availability is a very vital security requirement of mobile agent systems. Agent host must ensure the availability of data and services for the remote and local agents. It must be capable of detecting problems either in the form of software or hardware and resolve deadlock when problem arose. In mobile agent communication, agent platform must support ease of use of data and resources at both remote and local agents. For the agent platform to cope with the demands of mobile agents' communication, it must possess the following characteristics [19]:

a. Deadlock management
b. Concurrent access
c. Concurrency control
d. Restricted access
e. Capacity for agent dispatching
f. Ability to make available sharable data in a usable form
g. Capacity to notify agents of its inability to provide needed quality of service due to its capacity limitations rather than creating accidental denial-of-service for the starving agents.

2.5 Non-repudiation

In an agent system, repudiation occurs when an agent that participated in a transaction or communication later claimed that the transaction or communication never took place. An agent system must ensure that proper measures are in place to prevent non-repudiation among communicating entities. Whether the cause of repudiation is intentional or not, it can lead to serious reputation damage that may not be easily resolved. An agent platform may not be able to prevent an agent from repudiating a transaction or communication but it can ensure the availability of strong evidence to support the resolution of disagreements. The evidence may deter the agent that values its reputation and the level of trust other agents place in it from falsely repudiating future transactions or communications.

3REQUIREMENTS FOR MOBILE AGENT SECURITY

The following are the basic requirement for the security of mobile agents in a distributed environment [23]:

a. Agent authentication and authorization: the source and integrity of mobile agents should be ascertained, and access of a mobile agent to host resources should be subjected to an authorization check to verify its permission status.
b. Situatedness: A mobile agent should have knowledge of the environment it is executing, and be able to apply the needed security controls.
c. Autonomy and migration: An agent should have control over its migration and internal state and resist any unwanted interference. Higher degrees of autonomy and more cultured migration capabilities require greater levels of security as a result of the increased risks arising from agent code manipulation.
d. Communication: Communication of a mobile agent with its environment (i.e. other agents, hosts or humans) needs to be protected. Confidentiality, integrity, authenticity, non-repudiation and availability service requirements must be provided for the communicated data.
e. Rationality, veracity, and benevolence: The mobile agent should act in ahonourable way (and not act maliciously), expected by the execution environment and other agents.
f. Anonymity: The need for the identity of an agent may be paramount to some applications and services, while it may be necessary to hide such in other application and services for security reason.
g. Trust: For instance, with the use of reputation system, agents should be able to assess the trustworthiness of information received from another agent and from an agent platform.
h. Delegation: A mobile agent that intends to carry out certain tasks on behalf of another entity must be given permission provided the security of such delegation act supports the use of public key and attribute certificates.

4 SECURITY THREATS TO MOBILE AGENT IN OPEN MAS

Threats to mobile agent security are classified into: agent-to-agent threats, agent-to-agent platform threats, agent platform-to-agent threats, and others-to-agent platform threats.

a. Agent-to-agent threats: These represent threats that enable agents to exploit security vulnerability of other agents to launch attack against them on agent platform. Protection against this form of attack is made by implementation of separate agents on agent host[24]. These threats encompass denial of service attack, repudiation, unauthorized access, and masquerading.

*b. Agent-to-platform:*This threat occurs when a malicious agent attacks an agent platform. The attack usually takes different forms such as masquerading, denial of service (DoS), and unauthorized access to the host resources. However, agent platform could be protected against these attacks in different ways such as fault isolation [25], access control to host resources [26, 27], digital signature [28], strong authentication [29, 30], and proof carrying code [31]

b. Agent-to-platform: In this case, a malicious agent is attacking an agent platform. This set of threats includes masquerading, denial of service and unauthorized access. An agent platform can be protected against these attacks in several ways, such as software fault isolation, proof-carrying code, access control, sandboxing, authentication, and digital signatures.

c. Platform-to-agent: This is an attack from a hostile agent platform against a mobile agent. This type of threat seems to be difficult to mitigate. This is simply because the agent platform is in full control of the agent while the agent is executing on the platform. However, some researchers [28, 20, 33, 7, 24,] have succeeded in proposing solutions to some of the security issues. This set of threats includes denial of service, repudiation, masquerading, alteration, spying, eavesdropping, seizure or agent migration delay, thieving, and killing the agent.

d. *Others-to-agent platform:* This is a situation in which external entities, including agents and agent platforms, threaten the security of an agent platform. This set of threats includes masquerading, denial of service, unauthorized access, and copy and replay.

5 SOLUTIONS TO MOBILE AGENT THREATS

A. Protection of Mobile Agent

The essence of agent protection is to preserve the confidentiality and integrity of the agent code and data against leakage and tampering by unauthorized agent or platform. Mechanisms that are employed for the protection of mobile agents are[13]: execution tracing, co-operating agent, environmental key generation, computing with encrypted function, code obfuscation, and partial result encapsulation.

- Execution Tracing: It provides the means of detecting possible misbehaviour of host platform. Each host platform running an agent creates cryptographic trace of an agent execution, which is stored by the host. The trace contains the line of code executed by the agent and the external values read by the agent. The tracing activity is reiterated for all hosts in the agent's itinerary path. Having suspected foul execution of the agent, the

agent owner may commence verification by requesting a complete agent execution trace from the first host (a) and simulate agent execution based on the information logged in the trace. The result of the simulation provides the intermediate state and identity of the next host platform in the agent's itinerary. The agent owner also request and collect its trace log (b) and proceed in this manner until all the trace logs in the host platforms in the agent's itinerary are collected. If after the trace verification, a discrepancy is observed at some point, it shows that a malicious host is detected. However, execution tracing has a number of limitations such as (1) the possibility of creating large size of logs especially if the agent has visit large number of hosts. (2) the agent owner has to wait until a suspicious result is observed before he could run verification process. (3) It is difficult to be used in the case of multi-threaded agents
[32]

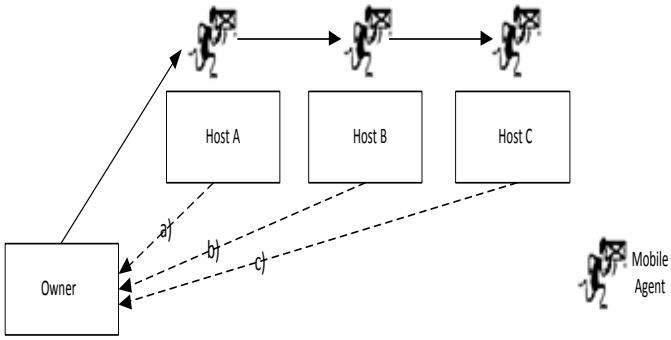

Fig. 1: Execution tracing – original protocol

- Co-operating Agents: With this technique, each agent accounts for and verify the route of its cooperating agent. Here also, critical task of an agent is distributed between the two co-operating mobile agents and they are made to execute the task in one or more separate sets of platforms [33]. Data are shared between the cooperating agents and they also exchange information thereby preventing shared data from being stolen by a single host[34, 35]. However, cooperating agents technique suffers some drawbacks such as (1) the cost of setting up communication channel for every migration. (2) It is difficult for a peer agent to decide which of the platforms is responsible in case a cooperating agent is maliciously killed.

- Environmental Key Generation: Riordan and Schneier [16] proposed environmental key generation which makes use of the keys generated from certain classes of environmental data. With these keys, mobile agents could receive ciphered messages that could only decipher if certain environmental conditions are true. Agents whose data and code ciphered using such keys could remain ignorant of their purpose until the environmental conditions are met. This technique of agent protection could be used in a situation when a sender communication with a receiver and the receiver could only receive the message after some conditions are met. The drawbacks with this technique are (1) the host platform of the incoming agent may be hostile to the agent. When the activation key is generated having met the required environmental conditions, the host platform could modify the agent to perform a different function, such as printing the agent code rather than running it. (2) The receiving platform may be skeptical to run the incoming agent since it is ignorant of the function of the agent. For example, the incoming agent could be a virus or contain some dangerous code.

- Computing with Encrypted Function: Sander and Tschudin [36] presents a software solution (cryptographic solution) approach to protecting mobile agent against malicious host platform during its hopping. The solution is focused at achieving privacy and integrity of mobile agent.In the solution suggested by Sander and Tschudin, the agent's home platform possesses an algorithm to compute a function (f) while the receiving platform has an input x. the receiving platform can compute f(x) as a service to the home platform. In fact, the home platform intends to blindfold the receiving platform about the significance of function (f). To accomplish this, the home platform encrypts the function (f) to obtain E(f). It then implement E(f) with the program P(E(f)) and embed P(E(f)) within the mobile agent before sending it to the receiving platform for execution. The receiving platform receives and runs the mobile agents including the execution of P(E(f)) at x to obtain P(E(f))(x). After the execution, the receiving platform sends the mobile agent back to the home platform which extracts the result from the mobile agent and decrypts it to get f(x). In fact, the receiving platform does not need to decrypt the encrypted program before executing it. This technique has offered safe execution of mobile agents on untrusted platforms. However, the main drawback is the difficulty of applying it to an arbitrary function (f). For now, the only classes of

functions on which encryption can be suitably applied are polynomial and rational functions [13].

- Code Obfuscation: Code obfuscation enables agent owners or creators to enforce security policy on agent code by applying behaviour preserving transformation to the agent code before the agent is sent to run on other platforms having some level of trust. Obfuscation prevents agent codes from being understood and possibly analyzed by foreign host [37]. In fact, the foreign host will not be able to modify the agent behaviour or breach the confidentiality of sensitive information hidden in its code. Such confidential information could be secret key, password, token, credit card number, or bidding limits. Layout obfuscation changes part of the agent code that are unnecessary to the execution of the agent code such as identifier names. Data obfuscation alters how data are stored in memory or how data are interpreted. Control obfuscation disguises the actual control flow in the agent code such as altering the ordering statement execution, changing the manner in which agent code statements are grouped together, hiding actual control flow behind irrelevant statements [38].However, the main challenge of this technique is the difficulty in implementation.

- Partial Result Encapsulation: This technique is used to detect tampering by malicious platform by encapsulating the results of an agent's execution at each platform in its travel path. Verification is subsequently carried out on arrival of the agent to its point of origination or also at intermediate points. Encapsulation could be done for providing confidentiality using encryption or to provide integrity and accountability using digital signature. The part of the agent encapsulated depends on the goals of the agent but mostly includes responses to inquiries made or results of transactions performed at the platform. There are three possible alternative means of encapsulating partial results (1) User provides way to encapsulate the information. (2) User relies on the agent platform capabilities. (3) User relies on trusted party for time-stamping a digital fingerprint of the results.

B. Protection of Agent Platform

Securing agent platform against unauthorized access to platform resources by mobile agent is inevitable to enforce the confidentiality of the resources. A number of mechanisms that have been employed to protect agent platform against attack by malicious mobile agent are*signed code, software-based fault isolation,state appraisal, proof carrying code, policy management, path history, and safe code interpretation.*

- Signed Code: This technique involves signing code or other objects with digital signature. A digital signature is used to confirm the originality or authenticity and integrity of an object. In most cases, the signer is either the agent creator/principal, agent user or some entities that are privileged to review the agent. As a result of the proxy activity of agent, agent systems commonly use signature of its principal to symbolize the authority under which the agent operates. Code signing involves public key cryptography which relies on key pair associated to an entity. One key is kept private with the entity while the other key is made public.

- Software Based Fault Isolation: This is a technique for isolating untrusted software module into a distinct fault domain enforced by software. The technique, commonly referred to as ***sandboxing***, allows untrusted program written in an unsafe language, such as C, to be safely executed within the single virtual address space of an application. Untrusted machine interpretable code modules are transformed so that all memory accesses are confined to code and data segments within their fault domain. Access to system resources can also be controlled through a unique identifier associated with each domain. The technique is highly efficient when compared with the use of hardware page tables to maintain separate address spaces for modules, when the modules communicate frequently among fault domains.

- State Appraisal: The focus of State Appraisal is to ensure that an agent has not been somehow disrupted due to modifications to its state information. The essence of this is to be sure that the agent has not become malicious by virtue of the modifications to its state. As a result of the possible changes to an agent state during its execution, hence it cannot be signed by the agent sender. The success of this technique is dependent on how far the harmful alterations to an agent's state can be predicted, and countermeasures taken by equipping the agent with appraisal functions before using it. Appraisal functions are used to define what privileges to grant an agent, based on conditional factors and whether

identified state invariants hold. An agent whose state violates an invariant cannot be granted privileges, while an agent whose state fails to meet some conditional factors may be granted a restricted set of privileges.

- Proof Carrying Code: The approach taken by proof carrying code[39] obligates the code producer (e.g., the author of an agent) to formally prove that the program possesses safety proper- ties previously stipulated by the code consumer (e.g., security policy of the agent platform). Proof Carrying Code is a prevention technique, while code signing is an authenticity and identification technique used to deter, but not prevent the execution of unsafe code. The code and proof are sent together to the code consumer where the safety properties can be verified. A safety predicate, representing the semantics of the program, is generated directly from the native code to ensure that the companion proof does in fact correspond to the code. The proof is structured in a way that makes it straightforward to verify without using cryptographic techniques or external assistance. Once verified, the code can run without further checking. Any attempts to tamper with either the code or the safety proof result in either a verification error or, if the verification succeeds, a safe code transformation.

- Policy Management: This is a protection mechanism that enforces local policy on host platform and is embedded within the agent. This policy framework offer several benefits when used to implement security such as reusability, extendable, verifiable, efficient and context sensitive.

- Path History: This technique provides the means to maintain a verifiable record of the platform previously visited by an agent to enable the newly visited platform to determine whether or not to process the agent and what resource constraints to apply. Computing a Path History requires every agent platform to add a signed entry to the migration path, indicating its identity and the identity of the next platform to visit, and to provide a complete Path History to the next platform. To prevent tampering, the signature of the new path entry must include the previous entry in the computation of the message digest. Upon receipt, the next platform can determine whether it trusts the previous agent platforms that the agent visited, either by simply reviewing the list of identities provided or by individually authenticating the signatures of each entry in the Path History to confirm identity. While the technique does not prevent a platform from behaving maliciously, it serves as a deterrent, since the platform's signed path entry cannot be repudiated. One obvious limitation is that path verification becomes more costly as the size of Path History increases. The technique also relies on the ability of a platform to judge correctly whether to trust the visited platforms identified.

- Safe Code Interpretation: Interpreted script or programming languages are often used to develop agents systems. The essence of this is to support agent platforms on heterogeneous computer systems. Java is one of the most widely used interpretive languages as a result of its support for code mobility, object serialization and remote method invocation, and other features that made it an ideal foundation for agent development. The idea behind safe code interpretation is that commands considered harmful can be either made safe for, or denied to, an agent. For instance, a good candidate for denial would be the command to execute an arbitrary string of data as a program segment. The limitations of safe code interpretation technique rest on the limitations of the interpretive language used to develop the agent system, example is the limitation of Java to account for memory, CPU, and network resources consumed by individual thread [40] and to support thread mobility [41].

6 CLASSIFICATIONS OF MOBILE AGENT ATTACK COUNTERMEASURES

Mobile agent attack countermeasures discussed above are categorized into *detection* and *prevention* as shown in Table 1.

Table 1: Classification of attack countermeasures

Attack countermeasures	Classification
Protected Entity	
Execution Tracing	Detection
Agent	
Co-operating Agents	Prevention
Agent	
Environmental Key Generation	Prevention
Agent	
Computing with Encrypted Function	Prevention
Agent	
Code Obfuscation	Prevention
Agent	
Partial Result Encapsulation	Detection
Agent	
Signed code	Detection

Agent Platform	
Software based fault isolation	Prevention
Agent Platform	
State appraisal	Detection
Agent Platform	
Proof Carrying Code	Prevention
Agent Platform	
Policy Management	Prevention
Agent Platform	
Path History	Detection
Agent Platform	
Safe code interpretation	Prevention
Agent Platform	

CONCLUSIONS

Much research has been focus on agent security in order to effectively deploy mobile agent based applications to overcome the problems associated to remote procedure call (RPC) characterized by the conventional applications. Mobile agents have found relevance in wide applications such as information retrieval, electronic commerce, brokering, smart grid, entertainment, robotics etc. They are often used to perform complex tasks such as continuous medical monitoring which are practically unachievable by human being. However, the security aspect of mobile agents has attracted enormous research efforts channeled towards preventing and/or detecting attacks on them.

In this paper we surveyed the main issues in the security of mobile agents. The security issue was viewed from two perspectives: *mobile agent* and *agent platform*. From the view of mobile agent, it was confirmed that it is much more difficult to ensure the security of mobile agents than the security of agent platforms. This is because, during the execution of mobile agent, it is fully under the control of agent platform. A malicious agent platform can easily modify, corrupt, steal its secret data or even kill the agent. However, integrating two or more of the aforementioned protection mechanisms and other approaches and ideas could evolve a comprehensive solution that would protect mobile agents. The ideal security solution for mobile agents can be provided by integrating host protection, agent protection, and user-to-agent trust techniques.

REFERENCES

1. Ahuja, P. and V. Sharma, *A Review on Mobile Agent Security.* International Journal of Recent Technology and Engineering (IJRTE), ISSN, 2012: p. 2277-3878.
2. Alfalayleh, M. and L. Brankovic. *An overview of security issues and techniques in mobile agents*. in *Communications and Multimedia Security*. 2005. Springer.
3. Kourouthanassis, P.E. and G.M. Giaglis, *Introduction to the special issue mobile commerce: the past, present, and future of mobile commerce research.* International Journal of Electronic Commerce, 2012. **16**(4): p. 5-18.
4. Camarinha-Matos, L.M., H. Afsarmanesh, and R. Rabelo, *E-business and Virtual Enterprises: managing business-to-business cooperation*. Vol. 56. 2013: Springer.
5. Karjoth, G., *Secure mobile agent-based merchant brokering in distributed marketplaces*. 2000: Springer.
6. Klusch, M., *Intelligent information agents: agent-based information discovery and management on the Internet*. 2012: Springer Science & Business Media.
7. Chang, Y.-S., C.-T. Yang, and Y.-C. Luo, *An Ontology based Agent Generation for Information Retrieval on Cloud Environment.* J. UCS, 2011. **17**(8): p. 1135-1160.
8. Su, C.-J. and C.-Y. Wu, *JADE implemented mobile multi-agent based, distributed information platform for pervasive health care monitoring.* Applied Soft Computing, 2011. **11**(1): p. 315-325.
9. Camarinha-Matos, L.M. and H. Afsarmanesh, *Virtual communities and elderly support.* Advances in Automation, Multimedia and Video Systems, and Modern Computer Science, 2001: p. 279-284.
10. Kargl, F., et al. *Smart reminder-personal assistance in a mobile computing environment*. in *Workshop on" Ad hoc Communications and Collaboration in Ubiquitous Computing Environments" at CSCW*. 2002.
11. Rasmusson, A. and S. Janson. *Personal security assistance for secure Internet commerce*. in *In New Security Paradigms' 96, ACM*. 1996. Citeseer.
12. Huynh, T.D., N.R. Jennings, and N. Shadbolt, *Developing an integrated trust and reputation model for open multi-agent systems.* 2004.
13. Ahila, S.S. and K. Shunmuganathan. *Overview of mobile agent security issues—Solutions*. in *Information Communication and Embedded Systems (ICICES), 2014 International Conference on*. 2014. IEEE.
14. Jung, Y., et al., *A survey of security issue in multi-agent systems.* Artificial Intelligence Review, 2012. **37**(3): p. 239-260.
15. Armoogum, S. and A. Caully, *Obfuscation Techniques for Mobile Agent code confidentiality.* Journal of E-Technology, 2010. **1**(2): p. 83-94.

16. Riordan, J. and B. Schneier, *Environmental key generation towards clueless agents*, in *Mobile agents and security*. 1998, Springer. p. 15-24.
17. Roth, V. and M. Jalali-Sohi, *Access control and key management for mobile agents.* Computers & Graphics, 1998. **22**(4): p. 457-461.
18. Ssekibuule, R., *Mobile Agent Security Against Malicious Platforms.* Cybernetics and Systems: An International Journal, 2010. **41**(7): p. 522-534.
19. Pandey, A., *Security on Mobile Agent Based Communication System.* 2007.
20. Farmer, W.M., J.D. Guttman, and V. Swarup. *Security for mobile agents: Authentication and state appraisal*. in *Computer Security—ESORICS 96*. 1996. Springer.
21. Cao, C. and J. Lu. *A path-history-sensitive access control model for mobile agent environment*. in *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*. 2005. IEEE.
22. Mitrovic, N. and U.A. Arribalzaga. *Mobile Agent security using Proxy-agents and Trusted domains*. in *2nd International Workshop of Security of Multiagent Systems (SEMAS'02) at 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 02), DFKI Research Report*. 2002.
23. Kalogridis, G., *Preemptive mobile code protection using spy agents*, 2011, University of London.
24. Mishra, A. and A. Choudhary, *Mobile Agent: Security Issues and Solution.* International Journal of Computer Technology and Electronics Engineering (IJCTEE) Volume, 2012. **2**.
25. Mao, Y., et al. *Software fault isolation with API integrity and multi-principal modules*. in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. 2011. ACM.
26. Zhou, L., V. Varadharajan, and M. Hitchens, *Cryptographic Role-Based Access Control for Secure Cloud Data Storage Systems*, in *Security, Privacy and Trust in Cloud Systems*. 2014, Springer. p. 313-344.
27. Reeja, S., *Role Based Access Control Mechanism In Cloud Computing Using Co–Operative Secondary Authorization Recycling Method.* International Journal of Emerging Technology and Advanced Engineering, 2012. **2**(10).
28. Madhulika, G. and C.S. Rao. *Generating digital signature using DNA coding*. in *Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2014*. 2015. Springer.
29. Cheon, S.-P., et al. *The scheme of 3-level authentication mechanism for preventing internal information leakage*. in *Digital Information and Communication Technology and it's Applications (DICTAP), 2014 Fourth International Conference on*. 2014. IEEE.
30. Shin, S., et al., *An effective authentication mechanism for ubiquitous collaboration in heterogeneous computing environment.* Peer-to-Peer Networking and Applications, 2014. **7**(4): p. 612-619.
31. Pirzadeh, H., D. Dubé, and A. Hamou-Lhadj, *An extended proof-carrying code framework for security enforcement*, in *Transactions on computational science XI*. 2010, Springer. p. 249-269.
32. Tan, H.K. and L. Moreau. *Extending execution tracing for mobile code security*. in *Proceedings of Second International Workshop on Security of Mobile MultiAgent Systems (SEMAS'2002)*. 2002.
33. Ouardani, A., S. Pierre, and H. Boucheneb, *A security protocol for mobile agents based upon the cooperation of sedentary agents.* Journal of Network and Computer Applications, 2007. **30**(3): p. 1228-1243.
34. Roth, V. *Secure recording of itineraries through cooperating agents*. in *ECOOP Workshops*. 1998. Citeseer.
35. Roth, V., *Mutual protection of co-operating agents*, in *Secure Internet Programming*. 1999, Springer. p. 275-285.
36. Sander, T. and C.F. Tschudin, *Protecting mobile agents against malicious hosts*, in *Mobile agents and security*. 1998, Springer. p. 44-60.
37. Armoogum, S. and A. Caully, *Obfuscation techniques for mobile agent code confidentiality.* Journal of Information & Systems Management, 2011. **1**(1): p. 83-94.
38. Low, D., *Protecting Java code via code obfuscation.* Crossroads, 1998. **4**(3): p. 21-23.
39. Lee, G.C.N.P. *Safe Kernel Extensions Without Run-Time Checking*. in *Symposium on Operating Systems Design and Implementation. ACM*. 1996.
40. Chess, D., et al., *Itinerant agents for mobile computing.* Personal Communications, IEEE, 1995. **2**(5): p. 34-49.
41. Gray, R.S., *Agent Tcl: A flexible and secure mobile-agent system.* 1997.

BIOGRAPHIES

Ogunnusi O.S, Principal Lecturer at The Federal Polytechnic, Ado-Ekiti. He has 18years of teaching experience. He is currently pursuing His PhD at Universiti of Technology Malaysia. His Research interests are Software Engineering and Computer Security. He has over 15 publications to his credit.

Ogunlola okunola is currently a senior lecturer at The Federal Polytechnic, Ado Ekiti Nigeria. His research interests include game design, software Engineering and Computer Security.