Mobile Agent Systems with Multiple Layers of Security

Preeti R. Hargunani
Divensi Inc,
Bellevue,
Washington.

Sulakshana A. Borsune
2,Flagstone,Apt.254
Irvine,
California

Aarti G. Ambekar
Assistant Professor
D.J.Sanghavi College of
Engg., Mumbai,India

ABSTRACT
In today's modern era of internet and intranet where distributed networking is spreading out its effects to every nook and corner of globe, it is essential to focus on both internet and intranet facilities. Mobile agent technology can be seen as a bright emerging technology which can be equally beneficial to internet applications as well as intranet applications. It is a piece of code that can move within a given network and act and make decisions autonomously on behalf of a user/application. For real time applications mobile agents can be used as data collection agents, action triggering agents. There can be a vast number of mobile agent applications. While mobile agent applications are evolving, the point of concern remains is the Security implementations on mobile agents.This paper gives extension of work on mobile agents by working on parallel processing mobile agents and by using Java language's one of the security feature.

Keywords
Mobile Agent Security, Mobile Agents, Socket connection, Hashing Techniques, Java Byte code, Parallel Processing.

1. INTRODUCTION
Any software code that can travel from one machine to another machine under any kind of topology carrying its data and state along with itself could be termed as Mobile Agent.Mobile Agent can resume its execution from the state it had left on previous machine/platform. Thus a mobile agent could be thought of as a virus program with no malicious intentions and that can act on behalf of its owner. In order to achieve these goals they communicate and interact with other agents, they exchange information and take back the results to the user. Such mobile agents are given certain goals and they try to achieve these goals according to their intelligence. Mobile agents also have the capability to interact with other mobile agents in order to achieve their goals. In real world, some of the areas where mobile agent technology is beneficial are for example; military is one such area that benefits a lot from Mobile agents especially in the time of wars. Main Server of the military base can send out the agents to different sites for collecting information like tasks assigned to the troops, their positioning, receiving orders etc. E-commerce is another area that widely makes use of mobile agents. For example-using mobile agents a shopping website can perform several tasks simultaneously like placing order fromvendors, performing transactions, advertising their products etc. Nowthe user will put his money in mobile-agent only when he is confident that his money is secure and the agents can be trusted which are dealing with his money or transmitting some secure information. This is one reason which instigates much research effort in security of mobile-agents which has its own benefits when used in these areas. Some of the existing mobile agent systems are: Telescript, JADE, Aglets, and Ajanta etc.

The mobile nature of mobile agents makes them travel independently in the network, making them susceptible to various kinds of attacks. Thus exposing the mobile agents systems to various kinds of limitations in the area of security.

The objective of this paper is to recognize all counter-measures taken to avoid the attacks caused by a malicious host. These attacks may include the stealing of private data like credit-card number or e-money which it uses to buy the items, or manipulation in the results of an agent that it generates on the remote host and finally to protect itself from manipulation or alteration of its code itself. The work done will give the implementation of security features on mobile agent platform and the mobile agent itself either by the use of cryptography methods to solve the basic attacks or to protect the agent from the bigger attacks it will make use of the combination of encryption and other techniques to solve the problem.

2. RELATED WORK
Mobile agent promise to deal more efficiently and elegantly with the dynamic, heterogeneous and open environment, which is today's Internet. A mobile agent/code is an active entity that can act within a distributed system on agent places on behalf of its user, following a given task. The first referred paper: MagicNET [1] gives the security architecture for the creation, classification and validation to achieve trusted mobile agents. This paper helps in implementing some of the security features/levels on the mobile agents to retain its identity which would be useful to detect malicious agent.

The threats in mobile code systems can be broadly classified as:

Malicious Mobile Agent: Threats that originate from an agent which attacks on another agent platform. This kind of attack can be protected in many ways e.g. sandboxing, software fault isolation, proof-carrying code, operating system access controls.

b. *Agent to Agent Attack*: An agent attacking another agent on the agent platform, protection against this attack has been made as agentseparation implemented on hosts.

c.*Malicious Host or platform*: An agent platform attacking an agent, protection of agents from malicious hosts remains a major problem in agent technology.

Major problem is how to secure an agent's private information which it is carrying along with it, the messages it is generating on the remote host and finally how to secure itself from unwanted alteration.

Another referred paper of GSMA [2] speaks on the concept of generating and distributing the sub-agents instead of actual mobile agents if the host platform is found to be malicious, thus helps in the fixing the problem of malicious host.Another paper [9] on "Mobile code security by java byte code instrumentation" speaks on the importance of java byte code usage as JVM's byte code verifier checks the byte code programs before execution

and a byte code interpreter performs run-time test. This paper speaks on the technique using a runtime tests into java code. Some of the techniques are class modification, involving the sub-classing non-final classes, and method-level modifications that may be used when control over objects from final classes is desired.

Another paper [10] on "SAS: a secure aglet server" speaks on some of the proposed mechanisms intended to protect both agents and hosts in order to foster the development of business applications that fully exploit the benefits of agent technology. These mechanisms laid the foundation for implementation of application specific protocols dotted with access control, secured communication and ability to detect tampering of agent data in Aglets system.The book referred of NIST publication [11] "Mobile Agent Security" details on the types of security threats and attacks, security requirements and the counter measures to give a full-proof mobile agent system.

3. PROPOSED METHODOLOGY FOR SECURE MAS

The focus of the paper is to build a secure mobile agent system by implementing security features on mobile agent who will be visiting different set of machines in the network and on the machines itself. This work deals with the agent systems, attacks on mobile agents and security. So it will be easier for the reader if s/he has the background knowledge of following technology:

a. Cryptography techniques.
b. Agent technology.
c. Any mobile agent system based on JAVA.

Mobile agent is a self-executing code, something similar to virus or a Trojan. The basic difference between the two is the fact that mobile agent concept comes with a lot of space for implementing security features, like to check each and every activity of agent as well as also security services could be applied to mobile agent itself and also to host platforms supporting mobile agents. Different types of securities applied on the system are mentioned in Table 1.This paper is concentrating on the mobile agent code and platform security.

3.1 Methodology

For implementation purpose an example of mobile agent application is consider, where two or more machines are connected in LAN. The purpose of such a setup is to enable one of the machines called as Initiator machine/host to send a set of files (for example: batch file, java file, etc.) on to the another machine known as remote machine/host for executing one of the batch files at the remote machine itself thus using the local resources of the remote machine.

One important thing to consider here is that any machine in LAN can act as initiator at any instant of time and start its transaction. Thus every machine in the connection is at equal level priority. Connections of the machines are as depicted in Fig.1.

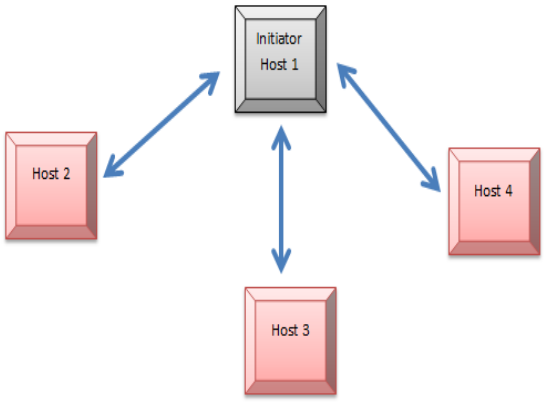

Fig.1: Connection of Remote hosts(host1, 2 and 3)and one of the Initiator Host

The system to be devised is thought on following lines of action and also as shown in Fig. 2:

A. Connecting the machines via socket interface.
B. Scan the remote machine and check whether the remote host is virus affected or not.
C. This scan report is sent back to initiator machine.
D. If the remote host is virus free then proceed for the next step.
E. Apply code signature on to the agent code on initiator machine.
F. Send mobile agent on to the remote machine along with the signature hash file for cross verification to remote host.
G. The received code is now further moved to next host i.e. host 2 for its execution on the host 2 platform.
H. This code is activated on the host2 machine after the necessary verification and results of the execution are sent back to host1.
I. Now Activate the mobile agent code after verifications performed at the remote host1 machine.
J. Once program executed, the results are stored in a text file.
K. The results obtained from host2 machine are merged with the results of host1 machine and are sent back to initiator machine.
L. The results in terms of a text file are shown on initiator machine.
M. Thus after returning back all the results to initiator, the processes on the remote host1 and host2 are forced to destroy themselves on the remote machines.
N. All these activities are maintained in a process log (on initiator machines, in our scenario its initiator machine and host1 machine which acts as initiator for host2 machine) for tracking the activity performed by the agent on its own platform and remote one.

From above mentioned points it is clear that the heart of this system is to devise a mobile agent and apply platform security and security on mobile code.

Table 1: Types of Security applied on the System

Sr. No.	Attacks	Description
1.	Security on	Before any mobile agent is sent on any machine, a check

	Platform	is conducted on the machine to know about their scan status.
2.	Security on Mobile Agents	A hash algorithm is applied on the mobile agent code so that before activating the mobile agent it could be verified.
3.	Self-termination on Mobile Agent	Mobile Agent self terminates or destroys itself on the host machine so that once the results are moved to initiator machine no more traces remain back at the Host machine.

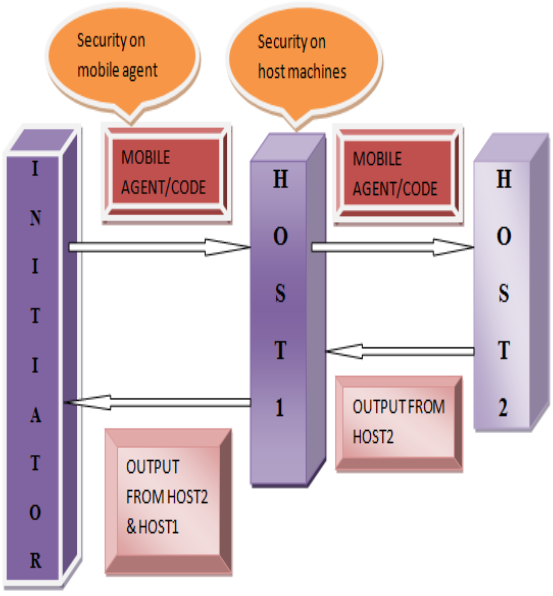

Fig. 2: Generalized View of the System

The above mentioned implementation points can be explained in details as follows:

Firstly a connection is needed to establish between initiator and remote host machines. To accomplish this, a socket interface is made between the two machines. For the implementation purpose local host as the IP address has to be passed.

Now the actual security implementation on designed mobile agent system can be done. Once the connections are established, the next step to be followed is to send the mobile agent code from initiator to the remote host.

This call for the execution of series of intermediate steps, i.e. Scan the host machine, send the scan report to initiator machine, calculation of message digest file for the JAR file to be sent, sending of batch file for executing the mobile agent, sending of JAR file along with message digest file to the remote host. These steps are explained below.
Mobile agent code to be sent within this application is a JAR file carrying .java file that is supposed to be executed at the remote host. This .java file is nothing but a file that would be executed at the remote host using its system CPU time to calculate the time required to complete a random task. Once this CPU time is calculated is stored back into a text file for further transfer purpose. This is the same file to be returned back to initiator machine for the verification purpose, i.e. this text file is returned back to initiator machine where it is executed to view its contents as shown in below Fig. 3.

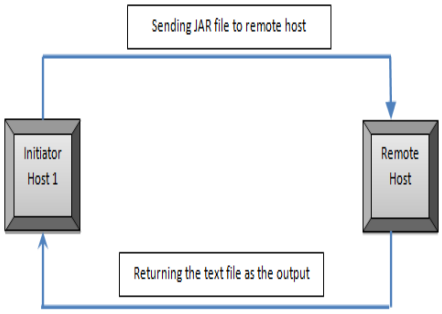

Fig. 3: Sending of JAR file from initiator to host and receiving of results (text file) from host to initiator.

First step to be followed here is, as soon as the mobile agent is ready to travel to its host, host machine should be scanned and the scanned report should be sent to initiator machine. Thus initiator machine can now take decision based on the received scan report. If the scan report says no virus the mobile agent can be sent to host machine. If else the scan report shows presence of virus in the system the communication is broken at the point only. The environment needs to be set at the remote host for the execution of jar file. Therefore before sending this jar file, three preliminary steps are supposed to be performed by the initiator.

A. To execute this jar file first initiator needs to send a batch file to remote host that would help execute the future jar file to be sent. This batch file contains the command line code to execute the .java file sent via jar file.

B. After sending the batch file, initiator has to ensure to that it calculates the hash file for the JAR file to be sent. This is required for the mobile agent validation by the remote host. Thus along with jar file, this calculated message digest file is also sent for verification purpose. This calls the actual start of remote host functioning. For generating a hash file we have used SHA method. SHA (Secure Hash Algorithm) is used because it is a set of cryptographic hash functions. A hash function is an algorithm that transforms (hashes) an arbitrary set of data elements, such as text file or a JAR file into a single fixed length value (the hash).This computed hash value may then be used to verify the integrity of copies of the original data without providing any means to derive said original data.

C. Once hash is calculated it is sent to remote host along with the JAR file. Thus as all the 3 files (namely, batch file, jar file, and message digest file) are received, host starts with its work.

a. First it calculates the message digest for the received JAR file.

b. This newly calculated message digest file is now compared with the received message digest file to confirm whether the received JAR file (which is actually the mobile agent code) is the valid one or tampered in between of the communication. For generating the message digest we are generating a .SHA file using SHA(Secure Hash Algorithm) encryption and decryption algorithm as described earlier.

c. The output of the comparison is flagged in terms of '0' and '1', where '0' indicates the outcome of comparison is

positive that is the received JAR file is a valid and verified (as shown in Fig. 4) , and '1' indicates the outcome of the comparison is negative that is the received JAR file is tampered one and not verified.

d. Outcome '0' specifies no tampering of original JAR file and thus permitting the process to move to next level in the program by sending a success intimation to the initiator as shown in the Fig. 4 and 5

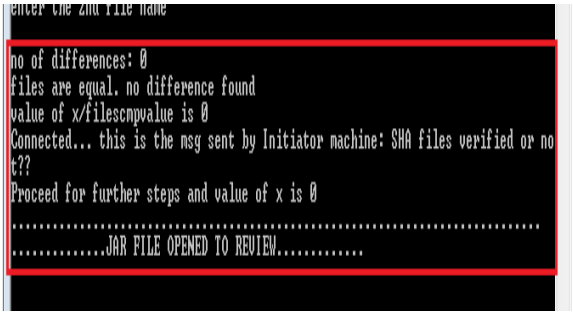

Fig.4: JAR verified on remote host and compare value '0'sent to initiator

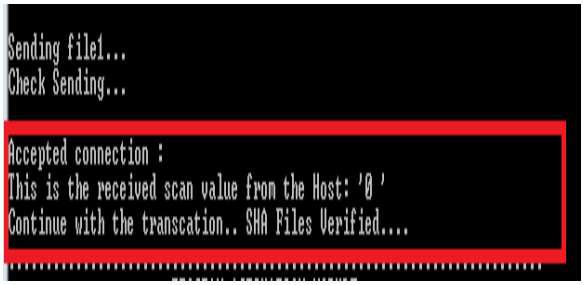

Fig.5: Compare value '0' received at initiator depictingthe JAR file verification as true

e. If the outcome is '1' then the connection between host and initiator is torn off and terminated abruptly as shown in one of the attacks which shows that if recalculated hash file is tampered one it results into comparison outcome as '1' thus tearing off the connections as shown in Fig. 6 & 7.

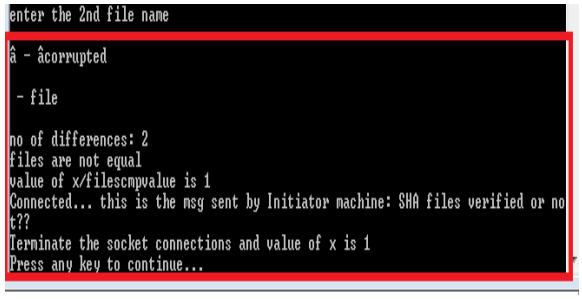

Fig.6: Hash Files are different thus JAR not verified, compare value '1' sent to initiator.

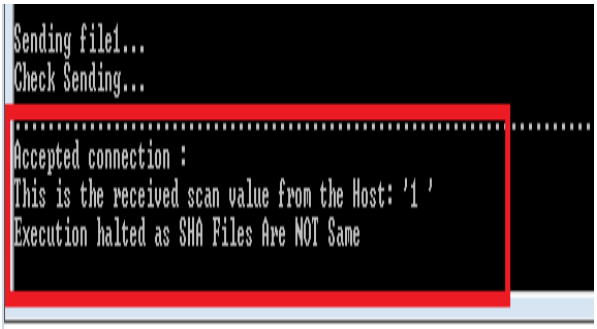

Fig.7: Compare value '1' sent to initiator depictingJAR file received at host is tampered and thusthe connection is terminated abruptly.

Once the JAR verification is positively done, next step is the Program activation (as shown in Fig.8(a) &8(b)) that means the remote host calling the batch file which in turn extracts the JAR file and executes the .java file from it and stores the output as a text file as shown in Fig 9(a) &9(b).

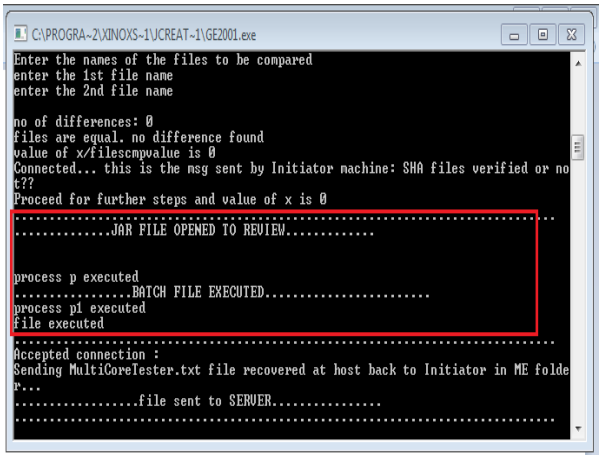

Fig.8(a): Compare value '1' sent to initiator depicting JAR file received at host is tampered and thusthe connection is terminated abruptly.

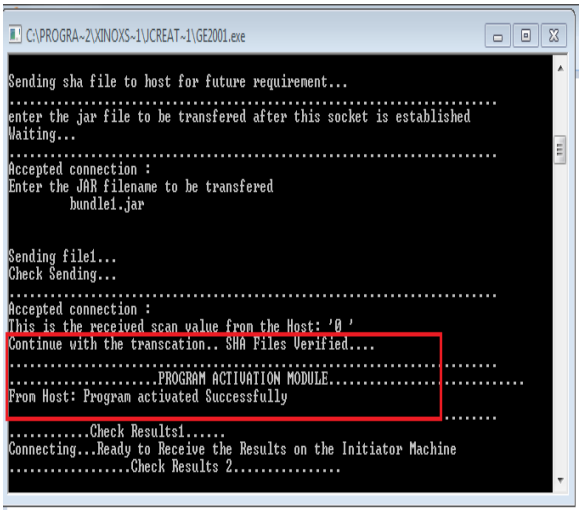

Fig.8(b): Indication of Program Activation display on Initiator

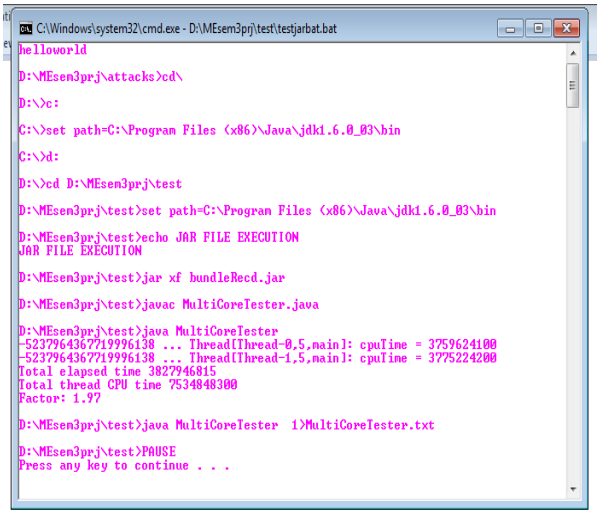

Fig.9 (a): Results displayed on console screen after program activation on remote host.

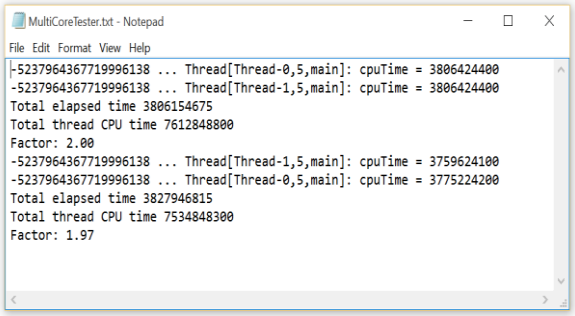

Fig. 9(b):Final results displayed on Initiator as text file.

Program activation on remote host results into the execution of a .java file which gives the system CPU time on console screen as shown in Fig. 9(a). The same .java file is again executed and this time the system CPU time is stored in .txt file which is to be sent onto the initiator machine for the review purpose. The control is now sent back to the initiator machine along with this text file. This text file as we have seen is nothing but the results returned back on the initiator machine after the execution of mobile agent on remote host. The text file is now executed to display the results recovered from host specifying the remote host time taken for completing the given task using the host machine's system time as shown in Fig 9(b).

Lastly, it is required that not even a single trace of the mobile agent code should remain back at the host platform. Hence after all the work done on both the sides it is required that all the processing done on the remote host machine should be removed. This call for the self-termination of all the files sent and executed on the remote host. Once all the above said work is done the ports held up during the overall implementation process for any kind of future use are to be made free. Thus all the ports used for the processing are closed and freed.

Up till now anything related to the documentation work. has not been mentioned .Documentation is one of the key features to be maintained. Documentation is one of the important set of deliverables after the overall process has been seen and executed. So for this project, all the phases i.e. the transaction between initiator and the remote host are maintained in terms of a log file namely "processLog.txt", maintained and updated on the initiator machine for any kind of references. For end users ease the overall progress of the execution in terms of phases noted in a text box area have been shown too.. Thus during the execution, every time all the details are displayed on the GUI text area as shown in Fig.10(a) and also all the execution details are stored along with the system date and time details in the log file on initiator machine as shown in Fig.10(b).

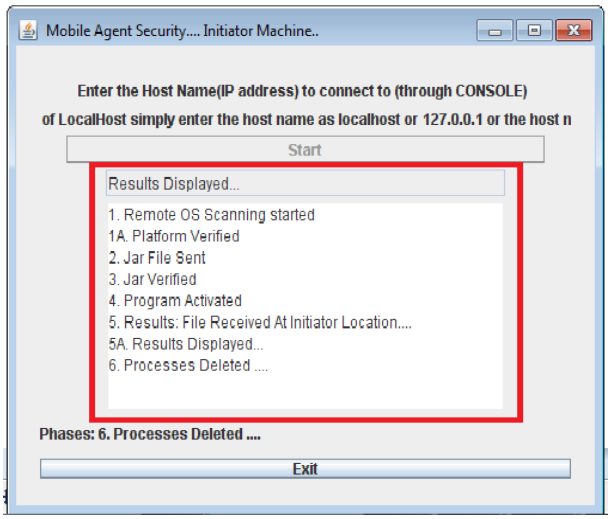

Fig. 10(a): Text Area

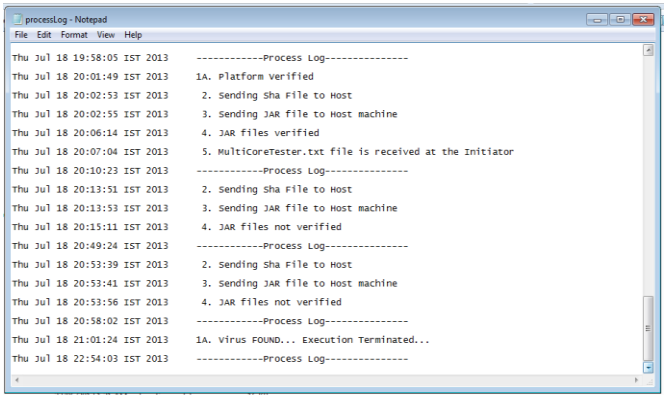

Fig. 10(a) Process Log

This ends the overall cycle of security implementation on mobile agent application.

4. CONCLUSION

Mobile agent system is one of the very promising paradigms; it has shown its presence in many applications like distributed networking, e-commerce applications etc. Still the benefits of mobile-agent technology cannot be fully exploited until all the security issues are properly addressed.The paper discussed about the mobile-agents, its implementation scenario and the security on mobile agent code. For implementation of a mobile agent system, socket programming is used to show the actual concept of mobile agents i.e. to carry code at the remote terminal and execute that code there itself and return with the results back to the initiator is seen.The proposed solution works for detecting and handling the security issues on mobile agent code by applying hashing algorithms on the mobile agent code before sending mobile agent. In case of this application, the end results returned to initiator in an ideal case (i.e. no attack case) is the CPU time of the host machine stored in a text file as shown in figure 9(b). The results show that the proposed security mechanisms are effective as they are capable of detecting as well as handling the said issue on

mobile agent code.The issues of security implementations like maintenance of process logs, applying cryptographic techniques to verify mobile agent and self-termination of mobile agents on its remote host are thus implemented using the simple socket interface and cryptographic techniques.

5. ACKNOWLEDGEMENTS

This is to thank eveyone who has contributed there work in this field. Special thanks goes to Prof. Irfan Siddavatam for his continous help and support to accomplish this task.

6. REFERENCES

[1] Muhammad AwaisShibli, SeadMuftic,*"MagicNET: Security Architecture for Creation, Classification, and Validation of Trusted Mobile Agents"*, ISBN 978-89-5519-139-4, ICACT 2009.

[2] Tarig Mohamed Ahmed, PhD, *"Generate Sub-Agent Mechanism to Protect Mobile Agent Privacy"*, IEEE 2012.

[3] *Joachim Baumann, Kurt Rothermel, "The Shadow Approach: An Orphan Detection Protocol for Mobile Agents"*, Second International Workshop, Germany, 1998.

[4] M. VigilsonPrem, S. Swamynathan, *"Securing Mobile Agent and its Platform from Passive Attack of Malicious Mobile Agents"*, *IEEE-International Conference on Advances in Engineering, Science and Management* (ICAESM -2012) March 30, 31, 2012.

[5] Danny B.Lange and Mitsuru Oshima, *"Programming and Developing Java Mobile Agents with Aglets"*. (Addison Wesley publication).

[6] T. Sander and C. F. Tschudin, *"Protecting Mobile Agents against Malicious Hosts"*. G. Vigna, editor, Mobile Agents and Security, volume 1419 of LNCS, pp. 44-60. Springer-Verlag, June 1998.

[7] F. Hohl, *"Time Limited Blackbox Security"*. G. Vigna, editor, Mobile Agents and Security, Volume 1419 of LNCS, pp. 97-102. Springer-Verlag, June 1998.

[8] William M. Farmer, Joshua D. Guttman, and Vipin Swarup, *"Security for Mobile Agents: Issues and Requirements"*, The MITRE Corporation, 202 Burlington Road, Bedford, MA 01730-1420

[9] Ajay Chander, John C. Mitchell, Insik Shin *"Mobile code security by java bytecode instrumentation"*.

[10] Evens Jean, Yu Jiao, Ali R. Hurson, Thomas E. Potok, *"SAS: A Secure Aglet Server"*, computer security conference 2007, April 11-13, 2007, USA.

[11] Wayne Jansen, Tom Karygiannis *"Mobile Agent Security"*, NIST Special Publication 800-19.

[12] *Kurt Rothermel, and Markus Schwehm, "Mobile Agents"*, Encyclopedia for Computer Science and Technology, New York: M. Dekker Inc., 1998.

[13] http://en.wikipedia.org/wiki/Mobile_agent

[14] http://www.davidreilly.com/topics/software_agents/mobile_agents/

[15] http://stackoverflow.com/questions/1741545/java-calculate-sha-256-hash-of-large-file-efficiently

[16] http://www.flexiprovider.de/examples/ExampleDigest.html