A Survey on Different Security Techniques of Mobile Code

Manoj Kumar Meena, Shweta Meena

Abstract- ***Mobile agents are software which moves autonomously through a computer network with aim to perform some computation or gather information on behalf of its creator or an application. In the last several years, mobile agents have proved its numerous applications including e-commerce. In most applications, the security of mobile agents is a burning issue. This article presents comparison of different aspects of mobile code security, namely the protection of hosts receiving a malicious mobile code and the protection of a mobile code within a malicious host.***

Keywords: Security, Mobile agents, Mobile code, malicious host, Electronic commerce.

I. INTRODUCTION

During the last several years, we have seen fundamental changes in distributed and client-server computer environment. This is due to the appearance of mobile agents (code) [1]. A mobile code is associated with at least two parties: its producer and its consumer. The mobile code paradigm encompasses programs that can be executed on one or several hosts other than the one that they originate from. Mobility of such programs implies some built-in capability for each piece of code to travel smoothly from one host to another. On the other hand, mobile agent technology has some limitations, primarily in the area of security. Possible vulnerabilities with mobile code fall in one of two categories: attacks performed by a mobile code against the remote host on which the program is executed and attacks performed by remote execution environment on mobile code.

II. MOBILE CODE SECURITY THREATS

When protecting a host from potentially malicious code, code mobility imposes the following security features:

1. Host and mobile code bear separate identities, therefore the mobile code's origin must be authenticated.
2. Mobile code is exposed through the network, hence the host must verify the integrity of the mobile code it just received.
3. The host does not generate the mobile code, but another party does: consequently, the actions it performs must be constrained through access control and/or checked through semantic verification.

Revised Manuscript Received on 30 March 2013.
* Correspondence Author
Prof. Manoj Kumar Meena, IT, IITM, Gwalior, (M.P.),India.
Prof. Shweta Meena, CSE, ITM, Gwalior, (M.P.), India.

When protecting a mobile code from a potentially malicious host, code mobility implies that the program will be run under total control of the host. This means the following threats:

1. Spoofing through impersonation of code owner.
2. Theft and secrecy violation through unauthorized disclosure.
3. Integrity violation through subversion of code semantics.

To prevent all three cases, data segments as well as code semantics must be protected.

III. DIFFERENT TECHNIQUES

3.1.Software Based Security

Advantages:

The low cost and compatibility with existing systems.

Limitations:

It require better techniques have to be invented due to evolving methods that circumvent application security. Furthermore, techniques to either secure either attack software are sped up by increasing computing power of processors and growing capacity of storage media.

3.2. Hardware Based Security: -

Advantages:

It provides much more speed up.

Limitations:

The high cost and the incompatibility with the base of current open computer platforms. By 'cost' we mean the expenses for buying and installing on the one hand and the cost for upgrading and maintenance on the other hand.

3.3. Client-Server Solutions:

Advantages:

It gives more control on mobile code.

Limitations:

The server or the network bandwidth becomes a bottleneck, causing services to be temporarily unaccessible. Although some extra overhead is needed to maintain communication between the client part and the server part. This directly indicates the main problem. At first glance this model seems to unload the server, nevertheless, in practice the client part and the server part have a highly interactive communication so that once more the bandwidth becomes a bottleneck.

3.4. Code signing:

The "Code Signing" technique ensures the integrity of the code downloaded from the Internet. It enables the platform to verify that the code has not been modified since it was signed by its creator. Code Signing cannot reveal what the code can do or guarantee that the code is in fact safe to run [9, 10]. Code Signing makes use of a digital signature and one-way hash function.

Advantages:

It works as a firewall so provides higher security for host against malicious code.

Limitations:

The disadvantage is that without extra security measures in place the code and the signature are still vulnerable to manipulation. If the signature scheme is known, one could simply change the code to its own needs, recomputed the signature and replace the old signature by the new one. The verification module would then just verify the new signature and would not suspect any tampering. The main reason for this vulnerability is that the signature and the verification module are not signed themselves.

3.5.Code Encryption:

Advantages:

Strength of security is directly proportional to strength of encryption function. It is best suitable technique for application which require high security.

Limitations:

During program execution parts of code will be decrypted 'on the y' with a secret key. Unfortunately, at that moment the code appears in the clear, in memory for example, so that it is able to intercept. The intercepted code can then be debugged, decompiled. This is de main vulnerability of this technique and furthermore makes the presence of a secret key this technique less suitable for distribution.

3.6. Code Obfuscation:

Obfuscation is a technique in which the mobile code producer enforces the security policy by applying a behavior-preserving transformation to the code before it sends it to run on different platforms that are trusted to various degrees [17]. Obfuscation aims to protect the code from being analysed and understood by the host. There are different useful obfuscating transformations [18, 21, 22]. Hohl [19] suggested using the Obfuscation technique to obtain a time limited black box agent that can be executed safely on a malicious platform for a certain period of time but not forever.

Advantages:

1. Low cost and the flexibility of this technique.
2. Depending on the need for security an application can be obfuscated accordingly. In their words: the extra cost and computation time, introduced by the transformations, can be specified by the software owner or programmer in advance.
3. Diversity: possibility to create different instances of one software application, to battle global attacks.
4. Low cost: low maintenance cost due to automation of the transformation process and compatibility with systems.
5. Platform independency: obfuscation can be done on high-level code so that platform independency is preserved.

Limitations:

1. Cost: every transformation introduce extra cost in memory and computation time necessary to execute the obfuscate program.
2. Security: obfuscation does not provide waterproof security.
3. Though, the only restriction for these transformations is preserving the functionality of the original program.
4. Furthermore, most of these code transformations are not one way and it is hard to decide where to use which transformations.
5. Nonetheless these techniques do not guarantee waterproof security, a combination of several transformation techniques can lead to sufficient practical protection against reverse-engineering and tampering attacks.

3.7. Sandboxing:-

Sandboxing [2] is a software technique used to protect mobile agent platform from malicious mobile agents. In an execution environment (platform), local code is executed with full permission and has access to crucial system resources. On the other hand, remote code, such as mobile agents and downloadable applets, is executed inside a restricted area called a "sandbox" [5, 6].

3.8. Proof-Carrying Code:-

Lee and Necula [19] introduced the *Proof-Carrying Code* (PCC) technique in which the code producer is required to provide a formal proof that the code complies with the security policy of the code consumer. The code producer sends the code together with the formal safety proof, sometimes called machine-checkable proof, to the code consumer. Upon receipt, the code consumer checks and verifies the safety proof of the incoming code by using a simple and fast proof checker. Depending on the result of the proof validation process, the code is proclaimed safe and consequently executed without any further checking, or it is rejected [2, 14, 15, 16].

3.9. White-box cryptography

Limitations:

So far the only practical disadvantages of white-box. cryptography are the code size and the extra execution time.

IV. CONCLUSION AND FUTURE WORK

In this paper we surveyed the main issues in the security of mobile agents. We considered both the mobile agent and the agent platform points of view, and reconfirmed that it is much more difficult to ensure the security of mobile agents than the security of agent platforms. We discussed the security threats and requirements that need to be met in order to alleviate those threats.

Table 1:Tabular/Graphical presentation of comparison of different techniques:-

Techniques	Protection against			
	Analysis		Tampering	
	static	dynamic	static	dynamic
Code Signing	N	N	F	N
Code Encryption	F	P	F	P
Code Obfuscation	P	P	P	P
White-Box Crypto	F	F	F	F
Tamper Resistant Soft.	F	P	F	F
Software Guard	N	N	P	N
Oblivious Hashing	N	N	P	P

N = None, P = Partial, F = (almost) Full

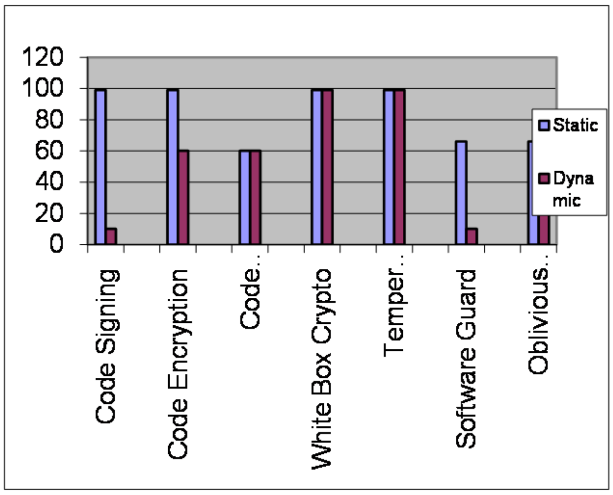

Figure:2 Protection against Analysis

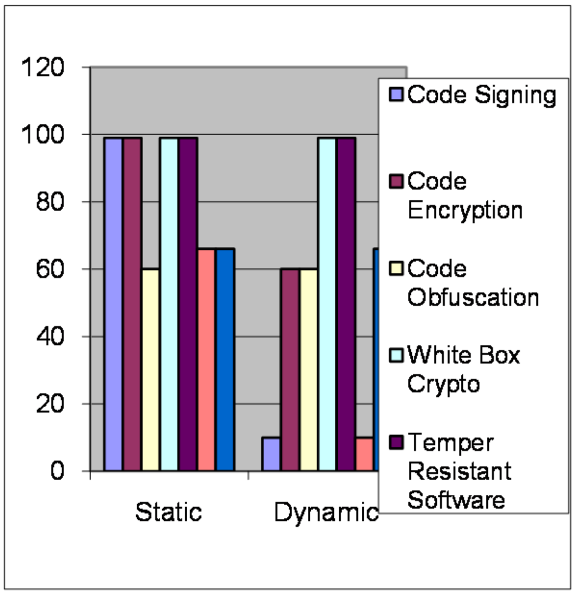

Figure: 3 Protection against Tampering

On the other hand, the protection of a mobile code against a malicious host is still an open research topic. Applying the theoretical solutions presented in this article to programming is far from trivial, and sometimes even unrealistic. Anyhow, it can be readily observed that non cryptographic techniques are generally not sufficient to protect a mobile code strongly.

REFERENCES

1. Software Security TechniquesInternal Report, COSIC], Jan Cappaert, Brecht Wyseur, and Bart Preneel KULeuven/ESAT/COSIC, October 2004
2. R. Wahbe, S. Lucco, T. E. Anderson, and S. L. Graham, “Efficient software-based fault isolation,” In Proceedings of the 14th ACM Symposium on Operating Systems Principles, pages 203--216, Dec. 1993.
3. D. Rubin and D. E. Geer, "Mobile code security," IEEE Internet Computing, 1998.
4. D. Chess, J. Morar, “Is Java still secure?,” IBM T.J. Watson Research Center, NY, 1998.
5. L. Gong, “Java Security Architecture (JDK1.2),” Technical Report, Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, California 94303, U.S.A, 1998.
6. Li Gong,”Secure java class loading,” IEEE Internet Computing, pages 56-61, 1998.
7. M. Hauswirth, C. Kerer, and R. Kurmanowytsch, “A secure execution framework for Java,” In Proceedings of the 7th ACM conference on computer and communications security (CCS 2000), pages 43--52, Athens, Greece, Nov. 2000.
8. L. Gong, M. Mueller, H. Prafullchandra, and R. Schemers, “Going Beyond the Sandbox: An Overview of the New Security Architecture in the Java Development Kit 1.2,” In Proceedings of the USENIX Symposium on Internet Technologies and Systems, Monterey, California, Dec. 1997.
9. “Signed Code,” (n.d.). Retrieved December 15, 2003, from James Madison University, IT Technical Services Web site: http://www.jmu.edu/computing/infosecurity/ engineering/issues/signedcode.shtml
10. “Introduction to Code Signing,” (n.d.). Retrieved December 15, 2003, from Microsoft Corporation, Microsoft Developer Network (MSDN) Web site: http://msdn.microsoft .
11. Gary McGraw and Edward Felten (1996-9). *Securing JAVA* [Electronic version]. John Wiley and Sons. http://www.securingjava.com/
12. M. Dageforde. (n.d.). “Security Features Overview,” Retrieved December 21, 2003, from Sun Microsystems, Inc. The JavaTM Tutorial Web site: http://java.sun.com /docs/books/tutorial/security1.2 /overview/
13. R. Levin (1998). “Security Grows Up: The Java 2 Platform,” Retrieved December 21, 2003, from Sun Microsystems, Inc. Sun Developer Network (SDN) Web site: http://java.sun.com/features/1998/11/jdk.security.html
14. P. Lee and G. Necula, “Research on Proof-Carrying Code on Mobile-Code Security,” In Proceedings of the Workshop on Foundations of Mobile Code Security, 1997.
15. S. Loureiro, R. Molva, and Y. Roudier, "Mobile Code Security," Institut Eurecom, 2001.
16. P. Lee. (n.d.), “Proof-carrying code,” Retrieved December 28, 2003, from Web site: http://www-2.cs.cmu.edu/~petel/papers/pcc/pcc.html
17. L. D'Anna, B. Matt, A. Reisse, T. Van Vleck, S. Schwab, and P. LeBlanc, “Self- Protecting Mobile Agents Obfuscation Report,” Report #03-015, Network Associates Laboratories, June 2003.
18. G. Wroblewski, “General Method of Program Code Obfuscation,” PhD Dissertation, Wroclaw University of Technology, Institute of Engineering Cybernetics, 2002, (under final revision).
19. F. Hohl, “Time Limited Blackbox Security: Protecting Mobile Agents from Malicious Hosts,” To appear in Mobile Agents and Security Book edited by Giovanni Vigna, published by Springer Verlag 1998.

*Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved*

20. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang, "On the (Im)possibility of Obfuscating Programs," in Advances in Cryptology, Proceedings of Crypto'2001, Lecture Notes in Computer Science, Vol. 2139, pages 1-18.
21. G. Hachez, "A Comparative Study of Software Protection Tools Suited for Ecommerce with Contributions to Software Watermarking and Smart Cards," Universite Catholique de Louvain, 2003.
22. C. Collberg, C. Thomborson, and D. Low, "A taxonomy of obfuscating transformations," Technical Report 148, Department of Computer Science, University of Auckland, July 1997.