International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-1 Issue-1, October 2011

A Survey on Different Security Techniques of Mobile Code

Sanjay Sharma, P. S. Patheja, Akhilesh A. Waoo, Rahul Gour

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. There are plenty of techniques to protect mobile code. There need a brief discussion about each method including strength and limitation so it may guide to choose best techniques for individual application. This paper presented a overview of various security techniques with their strength and limitation. 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.***

Key words: ***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) [8]. 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 id entities; 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 limited through access control and/or checked through semantic verification. 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

A. *Software Based Security*

Software can be protected in many ways. It can rely on trusted hardware, which is hardware based protection. Or it can rely on its own implementation and the underlying software, which is called software based protection. Some techniques combine both. For example, a in a layered security model, the critical applications can rely on a secure operating system, while the secure operation systems runs on top of trusted, cryptographic hardware.

Strength

The low cost and compatibility with existing systems.

Limitations

i.It requires 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.

B. *Hardware Based Security*

Hardware technique to protect software is tamper resistant packaging. In this case the software and data are physically shielded from attacks.

Strength

i. It provides much more speed up.

Limitations

i. The high cost and the incompatibility with the base of current open

Manuscript published on 30 October 2011.
* Correspondence Author (s)
Dr. Sanjay Sharma*, A.P, Department of MCA, MANIT, Bhopal (M.P.), India, (email: ssharma66@rediffmail.com)
P. S. Patheja, Research Scholar, MANIT, Bhopal (M.P.), India, (email: pspatheja@gmail.com)
Akhilesh A. Waoo, Research Scholar, MANIT, Bhopal (M.P.), India, (email: akhilesh_waoo@rediffmail.com)
Rahul Gour, M. Tech. Department of CSE, BIST, Bhopal (M.P.), India, (email: rahul.gour2009@gmail.com)

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.

ii. In general, hardware solutions offer less flexibility than software solutions.

C. Client-Server Solutions

Client-server environments that allow complex forms of distributed computing. Throughout this evolution limited forms of code mobility have existed: the earliest being remote job entry terminals used to submit programs to a central computer and the latest being Java applets downloaded from web servers into web browsers. The thin-client/server computing model involves connecting thin-client software or a thin-client hardware device with the server side using a highly efficient network protocol such as Citrix's ICA. The thin-client/server architecture enables 100 percent server-based processing, management, deployment, and support for mission-critical, productivity, Web-based, or other custom applications across any type of connection to any type of client hardware, regardless of platform. The client hardware can include Windows-based terminals, PCs, NetPCs, network computers, Apple Macintosh computers, or UNIX devices. Using the thin-client/server computing model, you won't need to purchase or upgrade hardware just to run the latest software--instead, you'll be able to let it comfortably evolve, leveraging your existing hardware, operating systems, software, networks, and standards. Thin-client/server computing extends the life of your computing infrastructure considerably.

Strength

i. It gives more control on mobile code.

ii. Very high processing speed.

Limitations

i. The server or the network bandwidth becomes a bottleneck, causing services to be temporarily inaccessible.

ii. 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.

D. 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.

Strength

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.

E. Code Encryption

This technique represents a software solution for protecting a mobile agent from a malicious executing platform during its itinerary [6]. This is a cryptographic solution to achieve integrity and privacy of the mobile agent. Protecting integrity means that the mobile agent is made safe against tampering by a malicious platform. Achieving privacy means that the mobile agent can conceal its program (code) when it is executed remotely in an un trusted environment.

Strength

i.Strength of security is directly proportional to strength of encryption function.

ii.It is best suitable technique for application which requires high security.

Limitations

i.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.

F. 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 [2]. Obfuscation aims to protect the code from being analyzed and understood by the host. There are different useful obfuscating transformations [3,18, 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.

Strength

i. Low cost and the flexibility of this technique.

ii. 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.

iii. Diversity: possibility to create different instances of one software application, to battle global attacks.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

iv. Low cost: low maintenance cost due to automation of the transformation process and compatibility with systems.

v. Platform independency: obfuscation can be done on high-level code so that platform independency is preserved.

Limitations

i. Cost: every transformation introduce extra cost in memory and computation time necessary to execute the obfuscate program.
ii. Security: obfuscation does not provide waterproof security.
iii. Though, the only restriction for these transformations is preserving the functionality of the original program.
iv. Furthermore, most of these code transformations are not one way and it is hard to decide where to use which transformations.
v. 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.

G. *Sandboxing*

Sandboxing [4] 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" [3,6,7].

H. *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].

Strength

i. Strength of security is directly proportional to strength of proof caring code.
ii. It is best suitable technique for application which requires moderate security.

Limitations

i. It is extra bourdon of caring code
ii. Its affects on performance
iii. Make processing slow

I. *White-box Cryptography*

White-box cryptography is aimed at protecting secret keys from being disclosed in a software implementation. In such a context, it is assumed that the attacker may also control the execution environment. This is in contrast with the more traditional security [5,12]. The white-box attack context (wbac), in contrast, contemplates threats which are far more severe. It assumes that:

Strength

i. Fully-privileged attack software shares a host with cryptographic software, having complete access to the implementation of algorithms;
ii. Dynamic execution (with instantiated cryptographic keys) can be observed.
iii. Internal algorithm details are completely visible and alterable at will.

Security requirements for wbac-resistance are greater than for resistance to gray-box attacks on smartcards. The wbac assumes the attacker has complete access to the implementation.

Limitations

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

J. *Black-box Cryptography*

In traditional black-box models (as in: black-box testing), one is restricted to observing input-output or external behavior of software. In the cryptographic context, progressive levels of black-box attacks are known. Passive attacks are re-stricter to observation only (e.g. known-plaintext attacks, exhaustive key search);active attacks may involve direct interaction (e.g. chosen-plaintext attacks);adaptive attacks may involve interaction which depends upon the outcome of previous interactions (e.g. chosen plaintext-cipher text attacks).True black-box attacks are generic and do not rely on knowing internal de-tails of an algorithm. More advanced attacks appear to be `black-box' at the time of execution, but in fact exploit knowledge of an algorithm's internal de-tails.

Strength

i. Low cost and the flexibility of this technique.
ii. Low maintenance

Limitations

i. Introduce extra cost in memory and computation time.
ii. Do not guarantee water proof security.

K. *Execution Tracing*

Execution Tracing enables detection of any possible misbehavior by a platform, that is, improper modification of the mobile agent code, state, and execution flow. This technique is based on cryptographic traces that are collected during an agent's

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

execution at different platforms. Traces are logs of the actions performed by a mobile agent during its lifetime. Execution Tracing enables an agent's owner to check the agent's execution history and see if it contains any unauthorized modifications done by a malicious platform. Each trace contains identifiers of all the statements performed on a particular platform. There is the need to retain a potentially large size and number of logs. Each platform chooses a verification server and that might encourage and facilitate a possible malicious collaboration between a platform and the server.

Strength

i. Give all information about path of code
ii. Help to analysis the performance of code in individual host.

Limitations

i. Extra cost for maintain logs.
ii. Logs are also vulnerable to change.
iii. Cost for access and analysis of all logs

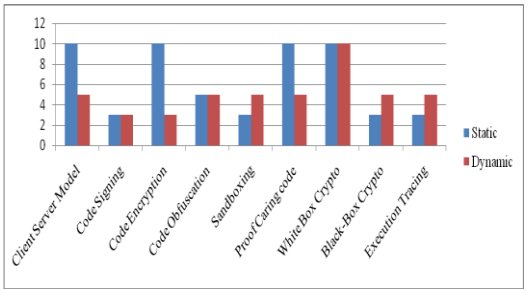

Fig 1.1 Protections against Analysis

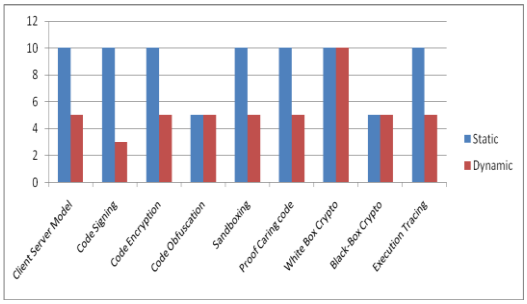

Fig 1.2 Protections against Tampering

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.

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. But on the other hand cryptographic techniques are slow and computational expensive. So we need select to a balanced method for much secure and high speed ended application.

References

1. Najmus Saqib Malik, David Ko and Harry H. Cheng, "A secure migration process for mobile agents", Published online 30 August 2010, U.S.A.
2. Wayne A. Jansen, "Countermeasures for Mobile Agent Security" March 01,2010
3. Sandhya Armoogum, Asvin Caully," Obfuscation Techniques for Mobile Agent code confidentiality", March 2010
4. AHMADI-BROOGHANI, ZAHRA, Proceedings of the 11th WSEAS International Conference on COMMUNICATIONS, Agios Nikolaos, Crete Island, Greece, July 26-28, 2007
5. Marc Joye,Thomson R&D France," On White-Box Cryptography" ,Published in A. El»ci, S.B. Ors, and B. Preneel, Eds, Security of Information and Networks, pp. 7{12,Tra®ord Publishing, 2008.
6. Li Gong,"Secure java class loading," IEEE Internet Computing, pages 56-61, 1998.
7. L. Gong, "Java Security Architecture (JDK1.2)," Technical Report, Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, California 94303, U.S.A, 1998.
8. Bo Chen1, Harry H. Cheng1, and Joe Palen2, SOFTWARE—PRACTICE AND EXPERIENCE published online, 13 July 2006,U.S.A
9. "Signed Code," (n.d.). Retrieved December 15, 2003, from James Madison University, IT Technical Services Web site:
10. http://www.jmu.edu/computing/infosecurity/ engineering/issues/signedcode.shtml
11. "Introduction to Code Signing," (n.d.). Retrieved December 15, 2003, from Microsoft Corporation, Microsoft Developer Network (MSDN) Web site: http://msdn.microsoft.
12. Robert Fischer, Ming-Yang Kao," Multi-Domain Sandboxing: An Overview" Sep. 2000
13. Chow, P. Eisen, H. Johnson, P.C. van Oorschot," White-Box Cryptography and an AES Implementation", Annual Workshopon Selected Areas in Cryptography (SAC'02), Aug. 15-16, 2002.
14. 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
15. 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.
16. S. Loureiro, R. Molva, and Y. Roudier, "Mobile Code Security," Institut Eurecom, 2001.
17. 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
18. D. Chess, J. Morar, "Is Java still secure?," IBM T.J. Watson Research Center, NY, 1998.
19. G. Wroblewski, "General Method of Program Code Obfuscation," PhD Dissertation, Wroclaw University of Technology, Institute of Engineering Cybernetics, 2002, (under final revision).
20. 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.
21. 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.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)

22. 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.
23. C. Collberg, C. Thomborson, and D. Low, “A taxonomy of obfuscating transformations,” Technical Report 148, Department of Computer Science, University of Auckland, July 1997.
24. 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.
25. Gary McGraw and Edward Felten (1996-9). *Securing JAVA* [Electronic version]. John Wiley and Sons. http://www.securingjava.com/
26. 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.
27. 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/