Mobile Code Security in Contemporary Information Systems - Past, Present and Trends

Denis Trček
Faculty of Computer and Information Science
University of Ljubljana
Tržaška cesta 25, 1000 Ljubljana, Slovenia - EU
denis.trcek@fri.uni-lj.si

Marko Bajec
Faculty of Computer and Information Science
University of Ljubljana
Tržaška cesta 25, 1000 Ljubljana, Slovenia - EU
marko.bajec@fri.uni-lj.si

***Abstract*—Despite the fact that mobile code security issues appeared some ten years ago, they remain important also in the era of service oriented architectures and cloud computing. Mobile code security certainly has some specifics, because it presents an opposite paradigm from the traditional one. In the traditional setting a host (i.e., local operating system, local environment) has to be protected against the code, while with the mobile code security this very code (more precisely, the program code and data) has to be protected against malicious host. And significant break-through is still to be seen in this area. Therefore this paper provides an analysis of the main problems related to mobile code security, and gives an outlook by identifying focuses of future research. It also provides a new paradigm called non-deterministic security services to address mobile code security issues. Clearly, even with the latest advancements in communications systems, the importance of secure mobile code remains one of the most challenging issues in information systems, and critical infrastructures.**

Keywords–information systems; critical infrastructures; distributed services; security; mobile code integrity.

I. INTRODUCTION

The latest advancements in information technology (IT) and information systems (IS) in general are leading towards SOA (services oriented architectures) [1], [2] and cloud computing [3] (actually, SOA is a concept that plays an important role in the area of cloud computing, and one concrete implementation of SOA are Web Services [4]).

Mobile code security issues were first tackled some ten years ago within the area of (intelligent) mobile agents. Despite the above new emerging IT and IS trends, these trends are not reducing the importance of mobile code and its security - one should just think of search engines and web crawlers that are needed for their operation.

The core problem of mobile code security is the opposite from the one in traditional security. While in traditional environments we want to protect local operating systems, programs and data from a malicious incoming code, in mobile code security area this paradigm is reversed. In this latter case we have to assure protection of mobile code, its original data and data that the mobile code has gathered while traversing the (global) network against a local, foreign malicious environment.

Many generic threats in mobile code security area have been identified in the past, and their overview will be given in the next section. In the third section, security mechanisms will be given that have been developed to counter these threats. In the fourth section, an analysis of the these counter measures will follow with the identification of open issues. In the fifth section, there will be argumentation that a paradigmatic shift is needed in the area of mobile code security, and the concept of non-deterministic security services will be given. There are conclusions given in the sixth section, which are followed by acknowledgement and references.

Last but not least, security services are about authentication, confidentiality, integrity, non-repudiation, access control, auditing and alarming [5]. The focus of this paper is on integrity of mobile code, which seems to be the hardest issue beside confidentiality.

II. MOBILE CODE SECURITY THREATS

The area of mobile code security threats is now quite reach, but it was comprehensively covered by NIST already at the end of the former century [6]. NIST work can be treated as a kind of a reference work and we summarize these threats as follows:

- Masquerade - A hosting platform may pretend to be the other one than claimed. In case of pretending to be a trusted third party, such platform may get access to confidential data, intervene with agent's communication with other agents, etc. Masquerade can also happen when an agent deploys services of a remote platform that is playing a masquerade.
- Denial of service - When a malicious hosting environment does not respond, it effectively stops the artificial life of an agent and prevents its goals and mission to be fulfilled. The platform may stuck the agent also by continuously assigning new subtasks to the agent. As agents communicate and may perform some distributed tasks, the whole community of agents gets stuck this way. It is also possible that the local platform (or remote

 ISBN: 978-1-61208-184-7

platform) that are needed by agent are attacked by some third party, and thus not able to provide services needed by an agent.
- Eavesdropping - The hosting malicious platform may not only eavesdrop an inter-agent communications, but all agent's data, its code and the execution process of this code. Even in case where parts of agent's data are encrypted, such platform has access to complete local life of the agent and can infer with the content of this encrypted code. Similar applies to agent's communication with other agents.
- Alteration - When being hosted by a malicious host, agent's code, state and data may be easily exposed to this host, and altered. The similar holds true for agent's communications. Let us state already at this point that the security service of (strong) integrity can prevent this threat to some extent, but it will be discussed in the next section how hard this is.
- Unauthorized access - Remote platforms and other agents may get access to agent's code, state and data, for which they are not authorized. In case of a local platform, this problem reduces to the above described threat of eavesdropping.
- Copy and replay - A malicious platform may simply reproduce an agent or its message(s), so an exact (and virtually regular copy) can be obtained. This can have serious security impacts - for example, original agent's messages may be reproduced by its clone. Therefore they can be accepted by other parties as fully valid, and reacted accordingly upon.

These threats have been quite successfully addressed in the latest years and will be described in the next section.

III. Mobile code security mechanisms and services

This section provides security mechanisms and services that were developed during the last years to address the above threats these are typical for mobile code security (beside from [6] below solutions are taken also from [7] and [8]):

- Co-operating agents principle - This principle requires that tasks are split and assigned to agents that run on different platforms (these agents should never encounter the same host), while these agents use authenticated (and confidential) channels to communicate.
- Execution tracing principle - This principle is intended to detect improper modification of agents code, state or execution flow. Cryptographic traces (logs of agents actions performed during its lifetime) serve for this purpose. Each platform produces digitally signed traces and appends them to the agent (digitally signed knapsack).
- Environmental key-generation principle - An agent waits until it receives a certain message (typically containing a decryption key) that is generated by other party when a certain condition is met in the environment. After receipt the agent decrypts its encrypted part of code and executes it.
- Code obfuscation - An agent is transformed in a way that preserves the intended behavior of the agent, while makes analysis of its code harder.
- Partial result encapsulation - With this technique the results of agent's calculations are encrypted by using the private key of a visited platform. Afterwards, the results are sent to domestic platform for verification.
- Sliding encryption - An agent uses a certain public key (e.g. of its domestic environment) to encrypt sensitive data, which can be later decrypted only by an entity that possesses the corresponding private key.
- Trail obscuring - With this technique an agent modifies its own binary image to make pattern matching harder so it cannot be identified as the same agent when traversing from one form to another.
- State appraisal functions - These functions serve to prevent tampering with agents dynamic data and reside in the encrypted (or digitally signed) part of the agent. They are built into agent by the sender (domestic environment) and can be used not only by the agent itself, but also trustfully behaving visited platforms.
- Encrypted functions (also referred to as mobile cryptography) - This approach deploys a principle where agent's code would be encrypted in a way that would enable correct execution at a guest platform, while the platform would not recognize the content (semantics) of this execution.

As to the last bullet - encrypted functions would really enable a significant advancement. The initial work in this area has been done by Sander and Tsudin in 1998 [9], and later extended by Lee, Alves-Foss and Harisson in 2004 [10]. The basic principle goes as follows [6]:

Suppose A knows an algorithm for computing a function f, while B has an input x for this function. A wants to compute $f(x)$ for B without revealing any details of f to B. If f can be encrypted in such way that it results in $E(f)$, then A creates program $program(E(f))$ and sends it to B. B executes $program(E(f))$ on x and returns the output to A who decrypts the result and obtains $f(x)$. This paradigm would provide effective means for solving mobile code challenges, but such cryptographic principle still needs appropriate concrete implementations (functions) to be found. Therefore things in this area (seem to) remain stalled at a theoretical level.

The last research that should bring some new advancement in this area is published in [11]. This solution is focused on confidentiality of code and agent's baggage together with integrity during agent's execution. The analysis of this paper reveals that the solution is about very complex architecture with many security protocols. History shows

 ISBN: 978-1-61208-184-7

that such complex structures are often leaking at some point (the formal verification of this solution is yet to be done). Further, the solution does provide integrity of a runtime code, but at a very high level through encryption of the whole code. Once a hosting platform is supposed to be trusted, it receives decryption key and from this point on, ”peeking & poking” of the code becomes feasible. This is not to say that this approach is useless - on the contrary. Our proposed solution actually further complements such solutions at a finer level and provides additional security, as will be seen in the rest of the paper.

IV. An analysis of existing countermeasures

In short - none of the above approaches radically reduces the problem of mobile code integrity and its protection against malicious host (except, of course, mobile code cryptography).

This is the core problem, which we will refer to as code morphing problem: When an agent is executed within a foreign operating system (or virtual machine) this operating system (or hypervisor in case of a virtual machine) can ”peek and poke” working memory locations (i.e., RAM locations) where agents execution code resides. The only 100% security would be if one could do strong integrity probing of executing agent in a RAM. But this is impossible because of operating systems principles like splitting a code and placing it into various RAM locations, loading it only partially into RAM (paging), etc. Therefore the actual binary image within various hosting environments can have too many varying digital representations when executed to provide a unique digital fingerprint.

Now what can be appropriate concept to counter this situation? The basic fact is that integrity is a security service and these services are implemented by using cryptographic protocols. This already gives the first hint, which (as a positive by product) complements research efforts at the level of security mechanisms (i.e., encrypted functions). Further, why shall one remain focused on a deterministic output at the level of security service? We could go for a solution where runs of a certain protocol with the same input would result in outputs that are dispersed according to some distribution on a certain interval. This distribution can then serve as a basis for calculation of a probability which of the obtained values is actually the correct one. And this is the core idea of non-deterministic security services.

Particular such approaches in this area already exist, starting with zero-knowledge (ZN) techniques on one side [12] and RFID tailored solutions on the other [13]. But they have not been recognized as a new concept so far. Such conceptual approach would be beneficial, and an advancement due to a paradigmatic shift would not happen for the first time in the area of security in computer systems (and cryptography in general). Actually, public key cryptography (PKI) was invented on the basis of a concept and principles

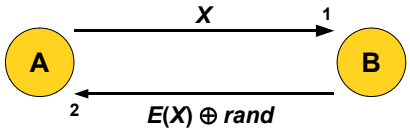

Figure 1. Classical challenge-response modified for non-deterministic security service (E stands for encryption function, X for challenge and $rand$ for a random bits string)

that were clearly developed in advance. PKI started with a strong focus of a freelance cryptographer W. Diffie on the problem of keys distribution. Diffie got in contact with M. Hellman at Stanford in the seventies of the former century. Having a clear concept in their minds they started to look for appropriate concrete candidate functions, and this led to the invention of public key cryptography.

V. Paradigmatic shift - non-deterministic security services

Now what is a non-deterministic security service? This kind of security refers to security services (i.e., cryptographic protocols) that

- do not provide a unique output after one run, so this output is from a certain interval of possible values and thus considered to be correct with a certain probability

OR

- require more (potentially parallel) runs with varying outputs, where all these outputs have to be considered as a whole to calculate the probability of assurance of the intended security service of the protocol.

To demonstrate the principle of the first case, assume a classical challenge-response authentication protocol, where the first message is, as usual, some random challenge. However, the second step is modified by EXOR-ing certain pre-agreed bit positions with a random string (see Fig.1). On receipt, the receiver should check all possible outputs as follows. All modified positions are initially assumed to be 0, and systematically changing them, bit by bit, one obtains outputs of all possible responses. If one of these messages produces the right output, the receiver can be assured about identity of the other party up to a certain probability (which depends on the number of masked bits and concrete encryption mechanism).

The first principle is given in [13], while to demonstrate the principle of the second case, let us consider Zero-knowledge based proof, the Fiat-Shamir protocol [14]:

1) Claimant A selects a secret s and computes $v = s^2 \bmod n$ and registers the result with a trusted center.
2) A sends to verifier B the value $x = r^2 \bmod n$.
3) Verifier B responds with a random $e \in 0, 1$.
4) A replies with $y = rs^e \bmod n$.
5) B verifies $y^2 \equiv xv^e (\mathrm{mod} n)$, and depending on e checks if the obtained value $y^2 = x$ or $y^2 = xv$.

 ISBN: 978-1-61208-184-7

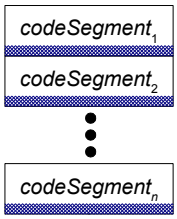

Figure 2. Agent's code structure (dashed areas denote segments' ICC, i.e., a code for its calculation and communication with the reference host)

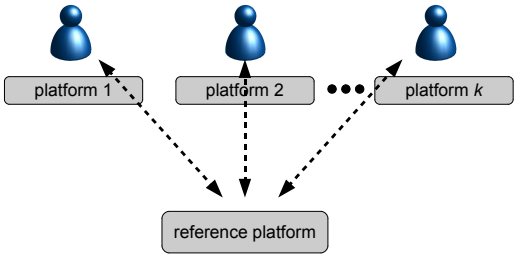

Figure 3. Protocol architecture for non-deterministic integrity checks

In the above scheme, a trusted center selects and publishes RSA-like modulus $n = pq$, where p and q are large primes that are kept secret, $s \in [1, n-1]$, and $y = 0$ is rejected, because it precludes $r = 0$. Without going into details, an attacker can impersonate A by choosing appropriate r, but once this value is selected, e in the second step defines the required result in the third step. As e is randomly chosen with probability 0.5, the attacker can successfully cheat in a single round with this probability. To reduce attacker's success probability, protocol can be run k times (steps 2 to 5), and the probability of successful cheating in this case becomes 2^{-k}.

Now how can non-deterministic security services be deployed for mobile code integrity? One possible protocol architecture is given in Fig. 2 and Fig. 3. The core premise of this approach is that each agent is segmented, and each of the segments has appended integrity check value (ICV). The main reasoning behind this is to actively check agents also during execution, when the code may be paged or dislocated in various segments of RAM.

Next, the agent is exactly replicated and forwarded to n subsequent hosts. Therefore these replicas are deployed over the network by communicating between a trusted, reference host, and n remote (distrusted) hosts. Remote hosts are continuously polled and the results are obtained and compared. As it is very unlikely that all hosts will be able to synchronize a coordinated attack on each instance and all related segments of its code, the reference platform will, in case of malicious host, obtain ICV values that will be dispersed. According to a concrete segmentation of an agent, number of its deployed instances, the nature of the polling protocol and the nature of the ICV, the reference platform is able to infer the probability of a proper behavior of a particular instance of the agent. Based on this data, reference platform can identify healthy agents and instruct them to proceed to other platforms, while attacked agents can be destroyed.

One complementary research already exists, but it is concerned with reliability of agents execution and deploys non-determinism in the agent code to improve this reliabilty [15]. So this work can be used for management of deploying multiple instances as mentioned above. However, t should be emphasized that the approach is not about security - it is about reliability and fault tolerance.

VI. Outlook and Conclusions

Mobile code and its security is playing a vital role also today in the era when internet has become a critical infrastructure and when it is architecturally heading towards cloud computing. However, as we have seen, one of the hardest issues is to ensure mobile code integrity. The paper has presented and analyzed approaches that were introduced during recent years, however, none of them radically reduces the problem.

Therefore this paper argues that we should change our paradigm and try to research mobile code integrity issues by deploying a new concept, called non-deterministic security services. This argumentation is based on the fact that paradigmatic shift was crucial also in the case of invention of public key cryptography (interestingly, paradigmatic shift is also visible in mobile cryptography area, too). With the proposed shift of paradigm the code is segmented and each segment checked for ICV during execution, while many replicas are deployed on various foreign hosts.

However, this solution requires development of new concrete implementations, where so-called lightweight protocols (and mechanisms) will play a central role.

Acknowledgments

Authors thank to Slovene Research Agency ARRS for support of this research with grant P2-0359 (Pervasive computing). We also thank to the reviewers that have provided constructive comments, which has helped us to improve the paper.

References

[1] Nagappan R., Skoczylas R., Sriganesh P.R., *Developing Java Web Services*, John Wiley & Sons, Indianapolis, 2003.

[2] Chase N., *XML Primer Plus*, SAMS Publishing, Indianapolis 2002.

[3] Mell P., Grance T., The NIST Definition of Cloud Computing, NIST Special Pub. 800 - 145 (Draft), Gaithersburg, 2011.

[4] Booth D., Haas H., McCabe F., Newcomer E., Champion M., Ferris C., Orchard D. (Eds.), Web Services Architecture, W3C Working Group Note, February 2004, http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/

 ISBN: 978-1-61208-184-7

[5] ISO, *Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 2: Security Architecture*, ISO standard 7498-2, Geneva, 1989.

[6] Jansen W., Karygiannis T., *Mobile Agent Security*, NIST Special Publication 800-19, Gaithersburg, 1999.

[7] Alfalayleh M., Brajkovic L., An Overview of Security Issues and Techniques in Mobile Agents, *Proc. of the Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS2004)*, pp. 59-78, Lake Windermere, 2004.

[8] Greenberg M.S., Byington J.C., Holding T., Harper D.G., Mobile Agents and Security, IEEE Communications Magazine, Vol. 36, No.7, pp. 76-85, IEEE, 1998.

[9] Sander T., Tschudin C., Protecting Mobile Agents Against Malicious Hosts. In G. Vigna, editor, Mobile agents and security, Lecture Notes in Computer Science, Vol. 1419, pp. 44–60. Springer-Verlag, 1998.

[10] Lee H., Alves-Foss J., Harrison S., The Use of Encrypted Functions for Mobile Agent Security, *Proc. of the 37.th Hawaii Int. Conference on System Sciences*, pp. 110, Hawaii, 2004.

[11] Shibli M.A., Muftic S., Giambruno A., Lioy A., MagicNET: Security System for Development, Validation and Adoption of Mobile Agents, *Proc. of the NSS 2009*, pp. 389-396, 2010.

[12] Goldreich O., Micali S., Wigderson A., *Proofs that yield nothing but their validity*, Journal of the ACM, Vol. 38, No. 3, pp. 690-728, 1991.

[13] Trcek D., Japinnen P., RFID security, RFID and sensor networks: architectures, protocols, security, and integrations, pp. 147–168, Taylor & Francis, Boca Raton, 2010.

[14] Menezes A.J., van Oorschot P., Vanstone S. A., Handbook of Applied Cryptography, CRC Press, Boca Raton, 1996.

[15] Mohindra A.,and Purakayastha A., Exploiting non-determinism for reliability of mobile agent systems, *Proc. of the International Conference on Dependable Systems and Networks*, pp. 144–153, New York, 2000.

 ISBN: 978-1-61208-184-7