Why Protection against Viruses, Bots, and Worms is so hard – Malware seen as Mobile Agents

Till Dörges
td@pre-secure.de
PRESECURE Consulting GmbH

May 31, 2007

Abstract: *Viruses, bots, worms, etc. are nothing else but mobile agents. Mobile agents in turn have been the scope of research in computer sciences for quite some years. Recently research on the security side of mobile agents has received increased attention, too.*

Perfectly securing mobile agents is generally impossible. While this is cumbersome for legitimate scenarios this is good news when trying to protect IT infrastructure. On the other hand, there are quite powerful protection methods for mobile agents so securing computers is far from trivial.

In order to explain this simple truth the paper relates current as well as well established findings from (theoretical) computer sciences to the IT security world of practitioners.

It is shown what methods are available to protect mobile agents, i. e. viruses, bots, and worms, from their environments, i. e. the computers they are running on. The limits of these protection methods are also explored.

Keywords: *Mobile Agents, Malware, Limits for Mobile Agents Security*

1 Introduction

This paper relates findings from mobile agents research to the world of malicious software or malware, i. e. viruses, bots, worms and the likes. It can also help in justifying why protection against modern malware is such a hard task. It is argued that perfect protection of malware is impossible. This generally is a positive thing but it has to be kept in mind that even without protection it might be hard to identify malicious code.

Section 2 will introduce the necessary definitions for mobile agent systems and explain why treating malware as mobile agents is valid. Section 3 will establish desirable security properties for mobile agent systems, and section 4 will discuss various results from research, showing the limits of mobile agents protection.

2 Mobile Agents and Malware

The definition of *agent* is subject to a lot of debate and no widely agreed upon definition seems to exist. It is, however, relatively safe to assume *autonomy*, *reactivity*, *proactivity*, and *goal orientation*. *Mobility* is not strictly required by agents but throughout this paper of course it is. Other features can include but are not limited to *social behavior* or the *ability to learn and adapt*. Usually agents exist and interact on *platforms*. Agents on a platform then constitute a *multi agent system* (MAS). For more details on agents and MAS see for example [Woo02].

Tightly coupled with agents and MAS is the paradigm of agent oriented software engineering (AOSE) which can be seen as an evolution of object oriented software engineering (OOSE). OOSE[1] and AOSE share modularization and encapsulation. AOSE adds features like autonomy, proactivity, and goal orientation. These can be particularly useful in complex and distributed scenarios where different protagonists / organizations – represented by agents – have to dynamically negotiate what steps to take.

For developing *multi agent applications*, i. e. applications built using (mobile) agents, several frameworks exists, e. g. JADE [BPR01], JACK [HRHL01], or MULAN/CAPA [DMR03].

Applications for agents range from simulations[2] to business process management (BPM) systems created with agent technology[3].

On the malware side many different forms of and names for malware exist. They include but are not limited to *virus*, *bot*, and *worm*. For a more detailed analysis and definitions see [Swi05]. The distinction has become rather blurry, as most malware of today has elements of each. Staniford et al. argue in [SPW02] that a virus requires some sort of user interaction for its propagation (making it similar to a *trojan horse*), whereas a bot can actively move from one computer system to another. Both exploit weaknesses (vulnerabilities, inadequate policies, inadequate user behavior, ...) in the systems they infect. A bot also requires some means of propagation (active or passive) and it additionally has to have some sort of command and control connection back to the miscreant on whose behalf it is acting. And of course all three can feature as many "services" (spy for information, attack other systems, back doors, ...) as wished by the miscreant responsible for their distribution.

While it might not be easy to characterize malware using these terms, it is very straight forward to examine prominent characteristics of today's malware:

Autonomy certainly is important for most malware because it means that it is able to run and execute without user intervention. In addition to autonomy many malwares employ stealth technologies (see [Rut07] for further details) to remain hidden from the person who is responsible for the infected system.

Reactivity is present for example, if the malware offers "services" like described above. The miscreant then can request certain actions and the malware reacts accordingly. This requires a control connection. A simpler type of reactivity would be to prepare the malware with a fixed set of rules and associated actions, e. g. to behave inconspicuously if an emulator or debugger is detected.

Proactivity means that the malware has a set of goals which it pursues proactively, e. g. propagation on to new systems or collecting information.

Mobility is needed by a malware if it wants to move from one computer system to the next.

Self Replication Often miscreants want to seize control over as many computer systems as possible (in order to build a *bot net*). This means that the malware has to create as many copies of itself as possible. Replication is usually combined with mobility, i. e. not the current malware itself moves to new platform but a copy instead. An interesting difference between malware and benign agents is that the latter often have a concept of identity and hence do not replicate, or if they do a new identity has to be created.

[1] This is not an extensive description of OOSE in any way. Inheritance for example is not regarded at all.

[2] For example the NASA uses a modeling and simulation tool called Brahms, http://is.arc.nasa.gov/HCC/tasks/Brahms.html.

[3] For example the Adaptive Enterprise^{TM} Solution Suite by Agentis, http://www.agentissoftware.com/.

Adaption Advanced malware is able to alter its behavior and its appearance throughout its life cycle. The former can be accomplished by updates through the control channel, e. g. bug fixes or new features, whereas the latter is achievable through polymorphic code.

Interestingly most current malware is autonomous, reactive, proactive, mobile, and self replicating. This makes it perfectly valid to investigate malware from the mobile agents perspective, much in the sense of Vigna who stated in [Vig04] that "mobile agents are suspiciously similar to worms".

Since an agent always runs on a platform, malware has to have something equivalent in order for the comparison to hold on this end, too. Originally platforms were conceived to provide a runtime environment for agents and standardized services for interaction between agents, transport, etc:

- An infected computer can easily provide for a runtime environment.
- Interaction between agents on a platform is less important from the malware perspective because one instance of a malware is enough to control the system. But if desired these features could be implemented by the agents directly, since interaction between agents mostly happens through message exchange. Besides message passing between the miscreant controlling the malware and the malware itself is present already. There also exist malwares which have the ability for peer to peer (P2P) communication.
- Mobility or explicit transport services are not provided by an infected computer system, either. But it is rather simple to integrate mobility into most malwares.

This implies that an infected computer system can be treated as a platform. Both the infected system and the malware then constitute a multi agent system.

It has to be noted, that *mobile code*, e. g. JAVA applets or ActiveX controls, does not necessarily qualify as a mobile agent. While it is mobile it usually lacks autonomy and proactivity. However, many of the discussions in the following sections apply to mobile code as well. For further details German readers are referred to [Eck04, pp. 67].

Last, but not least, example scenarios for both the good, i. e. mobile agent, and the bad, i. e. malware, perspective shall be given.

Example applications for the use of MAS based applications have been cited above. It is crucial to note that an application that was built using MAS techniques does not have to make up a MAS itself. The almost classical examples for mobile agents include the shopping agent, who scouts prizes on its owners behalf and possibly purchases items, and the database query agent, who searches one or more databases for its owner. Considering the computing power of mobile phones or PDAs today on one hand and the fact that they move on the other hand, this certainly gives another MAS like scenario worthwhile investigating.

Unfortunately occurrences of malware are a lot more frequent than those of benign agents. [BKH+06] for example explains how different kinds of malware can be collected and [BHKW05] explores both bots and bot nets. In order to fathom the proliferation of malware it might also be helpful to look on the economic side [TM06].

3 Desired Security Properties for Mobile Agent Systems

Security often is defined (see for example [Bis03, pp. 3]) in terms of

Confidentiality i. e. only those allowed can access resources,

Integrity i. e. only those allowed can alter resources, and

Availability resources must be accessible.

But in the general case this definition falls short, because the definition of security is almost innate for any specific system. Hence the above definition can only be the basis for further refinements.

For example two aspects that usually have to be considered when investigating security for multi agent system are *trust* and *identity*. This is simply because other actions, e. g. granting access to resources, are based on trust, which in turn can only be placed if the identity has been established first. Clearly identifying agents already has its challenges but attributing trust is even harder, in terms of how to quantify or even formalize[4]. For a general discussion of security Gasser [Gas88] and Bishop [Bis03] can safely be recommended.

Further interesting security properties in agent scenarios are *accountability*, *delegation*, *non repudiation*, *anonymity*, or *privacy*. For a general discussion about security in multi agent system see for example [JK00] or [Bor03].

As for confidentiality, integrity, and availability it also has to be defined to what precise aspects of mobile agents they should be applied:

Communication Integrity certainly applies, because no agent or agent owner wishes messages to be altered in transit. If privacy is a concern even confidentiality is required.

Mobility The thoughts are equivalent to communication. One could even treat agents as messages.

Agent execution Generally, a mobile agent without any trusted hardware of its own is completely dependent on the benevolence of the platform. The limitations to this are explored in section 4. The platform could read or modify private data or even alter the agent program code itself. Both possibilities are generally not wanted, for example when money is involved.

And availability of course is a desirable property for all aspects. Generally security in multi agent system systems can be broken down to two points of view:

Protection of the Platform Malicious agents or malware can cause harm to the platform. This may be as a disruption of operation (*denial of service*) or as reading or manipulation of data or program code.

Protection of the Agent An agent can either be harmed by a malicious platform or another malicious agent. The above discussion w. r. t. to mobility, communication, and agent execution applies.

Since the scope of this paper is on seeing malware as mobile agents, discussing the protection of the platform is not of prime importance. As for a benign mobile agent on one hand, in most cases possessing perfect protection against harm from its environment, i. e. platforms and other agents, seems worthwhile. In the malware case on the other hand perfect protection obviously is a rather discouraging perspective.

[4] This is dealt with by *assurance* [Bis03, pp. 475] and *dependability* [Dep07].

4 Security Measures for Mobile Agent Systems

The following chapter deals with the security measures available to protect mobile agents. On a short aside protection of the platform is briefly discussed. Key literature in this area certainly is provided by Jansen and Karygiannis in [JK00], where they provide an excellent overview about mobile agent security. [Vig98b] was published a bit earlier but is also important. A more recent thesis particularly focuses on communication aspects [Bor03].

4.1 Protecting the Platform

Protection of the execution environment against erroneous or malicious is a comparatively old discipline. Probably the most prominent ideas of how to protect a platform from malicious agents are, first the *reference monitor* concept introduced by Gasser in [Gas88, pp. 28], and second the *sandbox* idea as utilized for example inside the Java virtual machine. Other approaches include *signed code*[5], *state appraisal* [FGS96], *path histories* [CGH+95], and *proof carrying code* [NL98].

4.2 Protecting the Agent

Unlike protecting the platform protection of the agent itself is a rather recent branch of research. The following sections give an overview over the different approaches possible.

4.2.1 Trusted Hardware

Having the agent run on trusted hardware – at least partially – gives the best protection possible. Of course trusted hardware can be influenced from the outside, too, e. g. by influencing clock signals, voltage, etc. It is also not impossible to read information from trusted hardware, e. g. through timing attacks and other means. But it generally is a lot harder to tamper with hardware than it is with software.

Trusted hardware is one of the underlying concepts of Microsoft's approach to *trusted computing*: Next-Generation Secure Computing Base (NGSCB)[6]. The project, however, seems to have come to a standstill. One of the reasons certainly is that the tight link to Digital Rights Management (DRM, sometimes spelled out as Digital Restrictions Management) has been criticized quite strongly.

For the scenarios relevant to this paper trusted hardware is not of interest.

4.2.2 Policies

In certain environments it might simply be enough to simply prohibit malicious behavior by both agents and platforms. But as such policies like these are not enforceable through technical mechanisms. If at all policies have to be employed together with audit trails and logging (see 4.2.3).

4.2.3 Logging

This means to establish means to reconstruct what an agent did on what platform, and what the platform did to the agent. Logging just by itself most likely does not prevent any malicious

[5] See for example Microsoft Authenticode.

[6] http://www.microsoft.com/resources/ngscb/default.mspx

activity. But if the agents or their owners plan to interact more than once the ability to reliably reconstruct who did what, can help a great deal in minimizing malicious activity, if sanctions can be applied. Examples from literature can for example be found in [CGH+95] or [Vig98a].

Note that privacy can be an issue, which might be alleviated by the introduction of a trusted third party.

4.2.4 Cooperation

Cooperation between agents can mean either distribution of information or functionality or mere redundancy. [Rot98] and [Sch97] investigate different aspects.

4.2.5 Cryptography

The next class of approaches lies in the cryptographic realm. The main problem here is, how can a mobile agent execute cryptographic operations without the platform learning about them. One solution of course is trusted hardware (see 4.2.1).

Partial Result Encapsulation An overview is to be found in [JK00, pp. 19]. While the techniques vary slightly (see for example *Sliding Encryption* in [YY97] or *Partial Result Authentication Codes* in [Yee99]), the general idea is to provide the mobile agent with secure data storage, i. e. encrypting valuable data with a public key, e. g. the agent owner's key. This can be useful if a mobile agent is to visit several platforms and does not want to reveal data obtained on previous platforms. However, the data cannot be used without the secret key, and it also cannot be hidden from the current platform.

Computing with Encrypted Functions Sander and Tschudin [ST98] propose a scheme where an agent does not ask the platform to execute the function f but an encrypted version thereof, i. e. $enc(f)$. For a particular value x this means that the platform does not compute $f(x)$ and hence cannot learn its value, but it merely computes $enc(f(x))$. This of course does not prevent denial of service or replay attacks. It is also difficult to come up with suitable encryption schemes.

Undetachable Signatures are the only known application for computing with encrypted functions. They were also proposed by Sander and Tschudin [ST98]. The concept has been improved by [KBC00] and [LAFH04]. The idea is to bind certain restrictions to the agent's signature function. In the above example f would be the signature function of the agent, and *enc* would not only be the encryption function – hiding f from the platform – but it would also include the restrictions or constraints that the agent wishes to enforce. An alternative solution to undetachable signatures based on digital certificates was proposed by [Bor03, pp. 129].

Environmental Key Generation as described in [RS98] proposes to unlock code or data hidden inside the agent. The condition itself could be stored using a one-way-function, much like passwords are stored on modern operating systems. This approach ensures that the hidden code remains private just until it is about to be executed. However the code will still be available in clear to the platform before its execution.

Secure Communication Encryption of messages to provide confidentiality while they are in transit is easy using public key cryptography. Hiding the contents of the message from the platform is not possible, but proving its authenticity using undetachable signatures

is possible. Decryption of messages received requires either a secret or a shared key. Hiding these as well as the message contents from the platform is not possible.

For certain aspects in this section it was stated that hiding information from the platform is not possible. But as of now it is not clear whether for example other applications of computing with encrypted functions (apart from undetachable signatures) exist.

4.2.6 Code Obfuscation

Barak et al show in [BGI+01] that *perfect obfuscation*, i. e. perfect information hiding, is generally impossible.

But Varnovsky and Zakharov [VZ03] propose an approach which allows to obfuscate a program in such a way that an attacker can only understand a "negligibly small" amount of information. Hohl in [Hoh98] also introduces the term *Blackbox Security*. An agent, i. e. program and data, are transformed into a blackbox which achieves the same results as the original agent, but which is a lot harder to analyze and understand. Hohl also proposes a few *mess-up* or obfuscation algorithms.

Since obfuscation is not cryptography the main problem seem to be to determine the effectiveness of the measures.

5 Results and Conclusion

This paper introduced the concepts and research results of mobile agents and related them to malware. It was argued that malware can effectively be seen as mobile agents and hence most of the corresponding results are of interest when investigating malware. From the discussion in section 4 it can be concluded that perfect protection of mobile agents and hence malware is impossible – provided they do not run on (own) trusted hardware. Trusted hardware (4.2.1), policies (4.2.2), and logging (4.2.3), do not apply when investigating malware. But cooperation in the sense of redundancy (4.2.4) certainly does. This is also true for cryptographic measures (4.2.5) and obfuscation (4.2.6). While redundancy might not be so interesting from the point of view of this paper, the results about cryptographic measures and obfuscation definitely are. And most of the aspects addressed can be found in the wild already. Knowing that no mobile agent can perfectly obfuscate its code seems reassuring, but it must not be forgotten that interpreting this very code and actually recognizing malicious software, even when it is unprotected, is a hard problem. Because in the general case programs are not able to decide what the purpose of another program is (see [Tur36]). It also has to be kept in mind that "conventional" malware not sporting any advanced obfuscation or cryptographic technologies is still sufficient to yield sufficient results.

Further areas of interest are for example to conduct a survey among today's malware in order to determine what features are being used how frequently. Another interesting question is how malicious platforms could influence malware or perhaps convert benign agents into malware. Last, but not least it appears promising to apply the thoughts in this paper to immobile, i. e. stationary, agents. This would lead directly lead to an investigation of GRID like scenarios.

References

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *LNCS*, pages 1–18. Springer, 2001.

[BHKW05] Paul Bächer, Thorsten Holz, Markus Kötter, and Georg Wicherski. Know your Enemy: Tracking Botnets – Using honeynets to learn more about Bots. http://www.honeynet.org/papers/bots/, March 2005.

[Bis03] Matt Bishop. *Computer Security: Art and Science*. Addison Wesley, March 2003.

[BKH+06] Paul Bächer, Markus Kötter, Thorsten Holz, Maximillian Dornseif, and Felix C. Freiling. The nepenthes platform: An efficient approach to collect malware. In Diego Zamboni and Christopher Krügel, editors, *RAID*, volume 4219 of *LNCS*, pages 165–184. Springer, 2006.

[Bor03] Niklas Borselius. *Multi-Agent System Security for Mobile Communication*. PhD thesis, Royal Holloway, University of London, 2003.

[BPR01] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Developing multi-agent systems with a fipa-compliant agent framework. *Software, Practice & Experience*, 31(2):103–128, 2001.

[CGH+95] D. Chess, B. Grosof, C. Harrison, D. Levine, C. Parris, and G. Tsudik. Itinerant agents for mobile computing. *IEEE Personal Communications*, 2(5):34–49, 1995.

[Dep07] dependability.org – homepage. http://www.dependability.org/, 2007.

[DMR03] Michael Duvigneau, Daniel Moldt, and Heiko Rölke. Concurrent architecture for a multi-agent platform. In Fausto Giunchiglia, James Odell, and Gerhard Weiß, editors, *Agent-Oriented Software Engineering III. Third International Workshop, Agent-oriented Software Engineering (AOSE) 2002, Bologna, Italy, July 2002. Revised Papers and Invited Contributions*, volume 2585 of *LNCS*, pages 59–72. Springer, 2003.

[Eck04] Claudia Eckert. *IT-Sicherheit: Konzepte, Verfahren, Protokolle*. R. Oldenbourg Verlag, third revised and extended edition, 2004.

[FGS96] William M. Farmer, Joshua D. Guttman, and Vipin Swarup. Security for mobile agents: Authentication and state appraisal. In Elisa Bertino, Helmut Kurth, Giancarlo Martella, and Emilio Montolivo, editors, *ESORICS*, volume 1146 of *LNCS*, pages 118–130. Springer, 1996.

[Gas88] Morrie Gasser. *Building a secure computer system*. Van Nostrand Reinhold Co., New York, NY, USA, 1988.

[Hoh98] Fritz Hohl. Time limited blackbox security: Protecting mobile agents from malicious hosts. In Vigna [Vig98b], pages 92–113.

[HRHL01] Nick Howden, Ralph Rönnquist, Andrew Hodgson, and Andrew Lucas. JACK - Summary of an Agent Infrastructure. In *Fifth International Conference on Autonomous Agents*, Montreal, Canada, 2001.

[JK00] Wayne A. Jansen and Tom Karygiannis. NIST Special Publication 800-19 - Mobile Agent Security. National Institute of Standards and Technology, 2000.

[KBC00] Panayiotis Kotzanikolaou, Mike Burmester, and Vassilios Chrissikopoulos. Secure Transactions with Mobile Agents in Hostile Environments. In Ed Dawson, Andrew Clark, and Colin Boyd, editors, *ACISP*, volume 1841 of *LNCS*, pages 289–297. Springer, 2000.

[LAFH04] Hyungjick Lee, Jim Alves-Foss, and Scott Harrison. The construction of secure mobile agents via evaluating encrypted functions. *Web Intelligence and Agent Systems*, 2(1):1–19, 2004.

[NL98] George C. Necula and Peter Lee. Safe, untrusted agents using proof-carrying code. In Vigna [Vig98b], pages 61–91.

[Rot98] Volker Roth. Secure recording of itineraries through co-operating agents. In Serge Demeyer and Jan Bosch, editors, *ECOOP Workshops*, volume 1543 of *LNCS*, pages 297–298. Springer, 1998.

[RS98] James Riordan and Bruce Schneier. Environmental key generation towards clueless agents. In Vigna [Vig98b], pages 15–24.

[Rut07] Joanna Rutkowska. Introducing Stealth Malware Technology. In Christian Paulsen, editor, *Sicherheit in vernetzten Systemen. 14. DFN-CERT Workshop*, Hamburg, February 2007. DFN-CERT GmbH, Books on Demand GmbH.

[Sch97] Fred B. Schneider. Towards fault-tolerant and secure agentry. In Marios Mavronicolas and Philippas Tsigas, editors, *WDAG*, volume 1320 of *LNCS*, pages 1–14. Springer, 1997.

[SPW02] Stuart Staniford, Vern Paxson, and Nicholas Weaver. How to 0wn the Internet in your spare time. In *Proceedings of the 11th USENIX Security Symposium*, August 2002.

[ST98] Tomas Sander and Christian F. Tschudin. Protecting mobile agents against malicious hosts. In Vigna [Vig98b], pages 44–60.

[Swi05] Morton G. Swimmer. *Malware Intrusion Detection.* Books on Demand GmbH, 2005.

[TM06] Rob Thomas and Jerry Martin. The underground economy: Priceless. *;login: – The USENIX Magazine*, 31:7–16, December 2006. Team Cymru. http://usenix.org/publications/login/2006-12/openpdfs/cymru.pdf.

[Tur36] Alan Mathison Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42(2):230–265, 1936.

[Vig98a] Giovanni Vigna. Cryptographic traces for mobile agents. In *Mobile Agents and Security* [Vig98b], pages 137–153.

[Vig98b] Giovanni Vigna, editor. *Mobile Agents and Security*, volume 1419 of *LNCS*. Springer, 1998.

[Vig04] Giovanni Vigna. Mobile agents: Ten reasons for failure. In *Mobile Data Management*, pages 298–299. IEEE Computer Society, 2004.

[VZ03] Nikolay P. Varnovsky and Vladimir A. Zakharov. On the possibility of provably secure obfuscating programs. In Manfred Broy and Alexandre V. Zamulin, editors, *Ershov Memorial Conference*, volume 2890 of *LNCS*, pages 91–102. Springer, 2003.

[Woo02] Michael J. Wooldridge. *Introduction to MultiAgent Systems*. John Wiley & Sons, Inc, 2002.

[Yee99] B. S. Yee. A sanctuary for mobile agents. *Secure Internet Programming*, pages 261–273, 1999.

[YY97] Adam Young and Moti Yung. Encryption tools for mobile agents: Sliding encryption. In E. Biham, editor, *Fast Software Encryption (FSE 1997)*, number 1267 in LNCS, pages 230–241. Springer, 1997.