MAlSim – Mobile Agent Malware Simulator

Rafał Leszczyna, Igor Nai Fovino, Marcelo Masera
European Commission
Joint Research Centre
Institute for the Protection and Security of the Citizen
Via Enrico Fermi 2749
21027 Ispra (VA), Italy
rafal.leszczyna@jrc.it, igor.nai@jrc.it, marcelo.masera@jrc.it

ABSTRACT

One of the problems related to the simulation of attacks against critical infrastructures is the lack of adequate tools for the simulation of malicious software (*malware*). Malware attacks are the most frequent in the Internet and they pose a serious threat against critical networked infrastructures. To address this issue we developed Mobile Agent Malware Simulator (MAlSim). The framework uses the technology of mobile agents and it aims at simulation of various types of malicious software (viruses, worms, malicious mobile code). Moreover it can be flexibly deployed over computer network of an arbitrary information system.

Categories and Subject Descriptors

I.6.7 [**Simulation and Modelling**]: Simulation Support Systems; C.2.0 [**Computer-Communication Networks**]: General (security and protection); I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Security, Performance

Keywords

ACM proceedings, simulation, security, critical infrastructures, mobile agents, computer attacks, malware

1. INTRODUCTION

In our work we address the problem of computer security of critical networked infrastructures[1] including information systems, communication networks, electricity and other energy networks and water networks. We study their vulnerabilities, the potential malicious threats that might affect them, the related detrimental attacks, and the countermeasures that can be put in place for securing those systems. We develop instruments for a better understanding of the risks, for the qualitative and quantitative evaluation of the security issues, for the determination of the security condition of systems.

One of our studies concentrates on developing a systematic approach for the identification and assessment of security risk threats to information systems. The approach is based on the systematic planning, performance and description of experiments with simulations of attacks affecting control and supervision systems. We analyse the network of a critical infrastructure and on the basis of our observations we reconstruct it in our laboratory. In this configuration we implement attack scenarios. Then analyse results in order to evaluate impact of the attack, test robustness and identify countermeasures. The description, preparation, execution and results of the experiments will form the information source for trust cases i.e. documented bodies of evidence that provide demonstrable and valid arguments that a critical infrastructure is adequately safe and secure [2].

However we have encountered the problem of lack of software and methodology for simulation of *malware* – malicious software that run on a computer and make the system behaving in a way wanted by an attacker [3].

Malware can be categorised into the following families [3]:

- Viruses – which are self-replicating programs able to attach themselves to other programs (*host files*) such as executables, word processing documents and require human interaction to propagate.
- Worms – self-replicating programs autonomously (without human interaction) spreading across a network.
- Malicious mobile code – lightweight Javascript, VBScript, Java, or ActiveX programs that are downloaded from a remote system and executed locally with minimal or no user intervention.
- Backdoors – bypassing normal security controls to give an attacker access to a computer system.
- Trojan horses – disguising themselves as useful programs while masking hidden malicious purpose.
- User-level RootKits – replacing or modifying executable programs used by system administrators and users.

[1]Critical Infrastructures are defined as organisations or facilities of key importance to public interest whose failure or impairment could result in detrimental supply shortages, substantial disturbance to public order or similar dramatic impact [1]. Today most of critical infrastructures depend highly on the underlying communication networks.

SIMUTOOLS 2008, March 03-07, Marseille, France

DOI 10.4108/ICST.SIMUTOOLS2008.2942

- Kernel-level RootKits – manipulating the kernel of operating system to hide and create backdoors.
- Combination malware – combining techniques of other malware families.

Malware based attacks are the most frequent in the Internet and they pose a serious threat against critical networked infrastructures.

For answering this issue, we decided to develop a malware simulation tool with our own forces. In this way Mobile Agent Malware Simulator (MAlSim) was created. MAlSim is a software framework which aims at simulation of various malicious software in computer network of an arbitrary information system.

The paper is organised as follows: in Section 2 we present a brief overview of the related works. In Section 3 we describe MAlSim and the simulation environment, in which the framework is deployed. Finally, in Section 4 we present our conclusions.

2. RELATED WORK

As already mentioned we haven't been able to identify any compound frameworks for performing simulations of diverse types of malware. However there are documented studies on simulation of particular malware families such as computer viruses and worms.

The studies on virus simulation tools span between:

- *Educational simulators* i.e. programs demonstrating the effects of virus infection [4]. This group of programs include Virus Simulation Suite written in 1990 by Joe Hirst, which is a collection of executables, that 'simulate the visual and aural effects of some of the PC viruses' [5]. Another example is Virlab [6] from 1993, which simulates the spread of DOS computer viruses, and provides a course on virus prevention. (As it can be noticed, the programs are quite out of date, and today they would rather serve just as a historical reference.)
- *Anti-virus testing simulators* i.e. programs which are supposed to simulate viral activity, in order to test anti-virus programs without having to use real, potentially dangerous, viruses. Unfortunately, it seams that only one solution of this type was developed [4], namely Rosenthal Virus Simulator [7]. The simulator is a set of programs which provide 'safe and sterile, controlled test suites of sample virus programs', developed for 'evaluating anti-virus security measures without harm or contamination of the system' [7]. Again the applicability of the suite is limited since it was written ten years ago.

Concerning the simulation of worms, the prevalent work was done on developing mathematical models of worm propagation [8, 9, 10, 11], which base on epidemiological equations that describe spread of real-world diseases. The empirical approaches concentrated mainly on single-node worm spread simulators [12, 13, 14, 15], which are dedicated to run on one machine. Only few distributed worm simulations were implemented [16, 17, 18] but they approach modelling of worm propagation in the Internet and thus they don't respond our need for simulation tool allowing experiments in an arbitrary network of predefined topology.

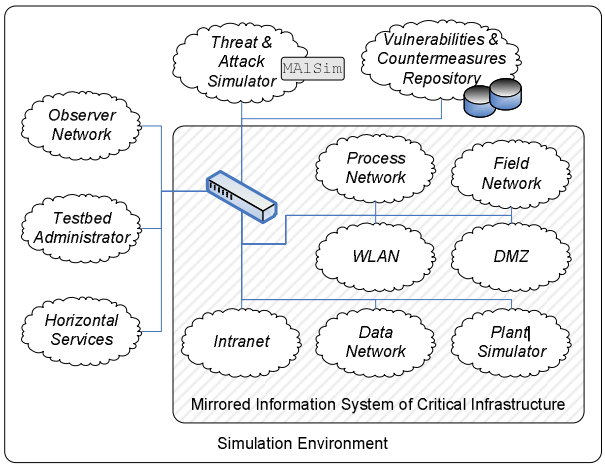

Figure 1: Simulation environment.

Also Trojan Simulator [19] has limited applicability in our studies. It was developed for evaluating effectiveness of anti-trojan software, and as such fulfills its purpose. However from the point of view of our experiments, it lacks the behavioural part, since the trojan malicious activities (e.g. stealthy task execution which consumes processor time or sending packets over network) are not simulated.

3. SIMULATION ENVIRONMENT

The simulations of malicious software are performed based on Mobile Agent Malware Simulator framework deployed in the simulation environment reconstructing the information system of an evaluated infrastructure (Mirrored Information System, see Figure 1). Additionally the environment comprises auxiliary parts which support configuration, performance and observation of the experiments; collection, storage and processing of results; and storage and provision of countermeasures.

3.1 MAlSim Framework

MAlSim – (Mobile Agent Malware Simulator) Framework is a software toolkit which aims at simulation of various malicious software in computer network of an arbitrary information system. The framework aims at reflecting the behaviours of various families of malware (worms, viruses, malicious mobile code etc.) and various species of malware belonging to the same family (e.g. macro viruses, metamorphic and polymorphic viruses etc.). The simulated software can refer to well-known malware (e.g. Code Red, Nimda, SQL Slammer) but also it can simulate generic behaviours (file sharing propagation, e-mail propagation) and non-existent configurations (which supports the experiments aiming at predicting the system behaviour in the face of new malware).

MAlSim Framework was developed using the technology of *mobile agents*. *Software agents* are software components, that are [20]:

- *Autonomous* – able to exercise control over their own actions.
- *Proactive* (or *goal-oriented* or *purposeful*) – goal oriented and able to accomplish goals without prompting

from a user, and reacting to changes in an environment.

- *Social* (or *socially able* or *communicative*) – able to communicate both with humans and other agents.

Mobile agents are software agents able to roam network freely, to spontaneously relocate themselves from one device to another.

Mobile Agent approach was chosen for the development of MAlSim because it particularly fits this purpose. Agents have much in common with malicious programs. Similarly to worms and viruses, they have the ability of relocating themselves from one computer to another. They are also autonomous as the worms are. At the same time they operate on agent platform which forms a type of sandbox facilitating their control.

Agent platform is an execution environment for agents which supplies the agents with various functionalities characteristic for the agent paradigm (such as agent intercommunication, agent autonomy, yellow pages, mobility etc.).

Agent platforms are deployed horizontally over multiple hardware devices through *containers*. On each device at least one container may be set up. Each container is an instance of a virtual machine and it forms a virtual agent network node. Containers make agent platform independent from underlying operating systems. Mobile agents are able to migrate from one container to another. Consequently, when containers are deployed on different devices, mobile agents can migrate between different devices.

Agent platforms can be imagined as agent communities where agents are managed and are given the means to interact (communicate and exchange services). Many agent communities may coexist at the same time. Depending on the implementation of the platform, agents may be able to leave one community (platform) and join another[2].

MAlSim is dedicated for the JADE (Java Agent DEvelopment Framework) agent platform.

JADE is a fully Java based agent platform which complies with the FIPA[3] specifications. It is provided by means of:

- Software framework which facilitates the implementation of multi-agent systems through a middleware which supports agent execution and offers various additional features (such as a Yellow Pages service or support for agents' mobility).

- Set of graphical tools that supports the debugging and deployment phases.

JADE is licensed under Lesser General Public License (LGPL), meaning that users can unlimitedly use both binaries and code of the platform. During over seven years of its development JADE has become very popular among the members of agent community and now it is probably the most often used agent platform. JADE is continuously developed, improved and maintained, not only by the developers from the Telecom Italia Lab (Tilab), where it was originated, but also by contributing JADE community members [30, 31].

Further details on the choice of JADE for our works can be found in [32].

[2]Further information on software agents an interested reader can find in [21, 22, 23, 24, 25, 26, 27, 28, 29].

[3]`www.fipa.org`

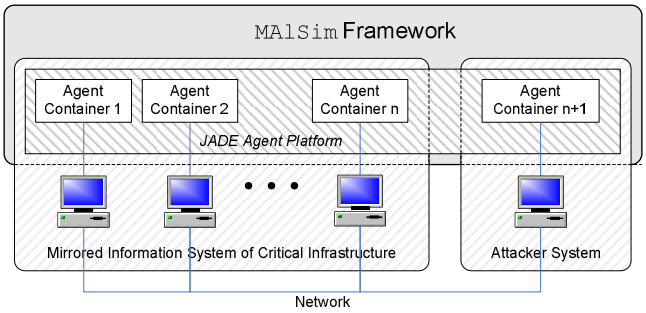

Figure 2: MAlSim deployment.

As shown in Figure 2 JADE has to be deployed over all hosts participating in the experiments with MAlSim. In our configuration these are the computers of Mirrored Information System and of the Attack and Threat Simulator. Java-based JADE is flexibly installable on various operating systems, and we deploy it on various distributions of Linux (Debian, Ubuntu, CentOS) and Microsoft Windows.

MAlSim Toolkit provides:

- Multiple classes of MAlSim agent (extensions of JADE `Agent` class).

- Various behavioural patterns implemented as agent behaviours[4] (extensions of `Behaviour` class).

- Diverse migration/replication patterns implemented as agent behaviours (extensions of `Behaviour` class).

The MAlSim agent class is the basic agent code which implements the standard agent functionalities related to its management on the agent platform, its communication skills and the characteristics related to the nature of simulated malicious software. This code will be propagated across the attacked machines.

To render it operative, the code must be extended with instances of the behaviour classes and the migration/replication patterns. Depending on the chosen behaviour(s) and the migration/replication patterns, the instances of the same agent class will be created on the attacked host, or instances of another agent class from the toolkit.

The behavioural patterns comprise actions performed by malicious software such as scanning for vulnerabilities of operating system, sending and receiving packets, verifying if certain conditions are met. To support the demonstrative aspect of experiments we have also developed patterns with audio-visual effects. For example, to facilitate the observation of malware diffusion in the network, a sound can be played by the agent after it arrived to a new container. The patterns also include operations such as disabling network adapter, enabling a local firewall to operate in all-block mode or starting a highly processor time consuming task etc. Using the patterns we can show the malicious effects of malware. For example that after malware infection, it is no longer possible to connect to the host, or that the host's performance is affected etc.

[4]In agents terminology the agent's *behaviour* is a set of actions performed in order to achieve the goal. It represents a task that an agent can perform [33].

Migration and replication patterns describe the ways in which MAlSim agent migrates across the attacked hosts. The patterns implement malware propagation models and the propagation schemas specific to a particular information system mirrored in the laboratory, which allow for characteristics such as: which networks of the system will be affected, in which order, at what relative time etc.

Currently our repository of agent classes and behaviours contains only basic malware implementations for zero-day viruses and worms, and we are planning to extend it in the foreseeable future.

As it was depicted in Section 1 malicious software migrate from one computer to another using network connections or portable data storage. They infect files (e.g., executables, word processing documents, etc.) or consist of lightweight programs that are downloaded from a remote system and executed locally with minimal or no user intervention (typically written in Javascript, VBScript, Java, or ActiveX). MAlSim on the other hand uses the migration mechanisms embedded in the agent platform.

In the default configuration (used for the MAlSim implementation) these mechanisms are realised over Java Remote Method Invocation protocol on port 1099. This introduces a difference between the simulated conditions and the conditions of simulation. Thus we are going to develop agent behaviours aiming at minimising this difference. One solution could be for example to not allow MAlSim agent migrate until a transport channel used by the prototype malware was opened. As a result, MAlSim agent, even if 'physically' moving through the connection on 1099 port, will behave as relocating through a HTTP or POP3 connection etc.

During MAlSim setup we take the following steps:

1. We withdraw the attack scenario from repository. An attack scenario is a sequence of steps taken during attack.

2. According to the chosen scenario we select the appropriate MAlSim agent from the database and configure it. If none of existing MAlSim agents fit the attack scenario, we develop a new MAlSim agent.

3. We extend the agent with migration schema (through adding agent behaviours from the repository).

4. We extend the agent with malicious behaviour.

The experiment are controlled through the graphical interface of the JADE main container installed on a PC of Attack and Threat Simulator. As shown in Figure 3 the graphical console allows also observation of the diffusion of the simulated malware.

3.2 Simulation Lab Environment

Malware propagation features depend heavily on the characteristics of the underlying network [17]. In contrast to the alternative works on malware simulation which use modelling approaches to reconstruct the underlying network, we build a real, physical configuration based on hardware of our cybersecurity laboratory.

Since our studies are focused on the security of critical networked infrastructures we reconstruct the situation occurring in an analysed critical infrastructure. For example, for the infrastructure of a power plant we mirror the process network (interconnecting diverse subsystems of the energy

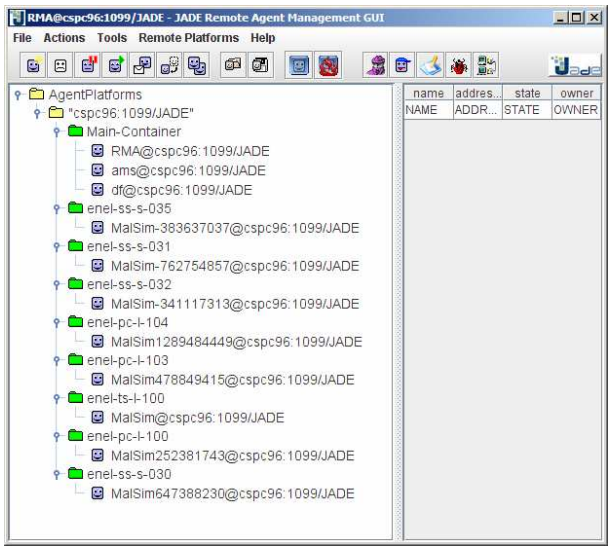

Figure 3: MAlSim Framework takes advantage of JADE GUI for control and observation of experiments.

production process), the field network (interconnecting controllers and field devices), the corporate network etc. (see Figure 1).

However it must be noted that MAlSim Framework is not stiffly fixed to the setting of our cybersecurity laboratory and can be easily installed in any simulation environment.

Configuration, performance and observation of experiments, collection, storage and processing of results and other functionalities supporting performance of experiments are provided by the following sections of our simulation environment:

- *Threat and Attack Simulator*, which aims at providing conditions for reconstructing attacks and threats that can jeopardise the analysed information system. It is the part of simulation environment where the simulated attacks are configured and launched. Since there are various and diverse attacks, when designing this part of the simulation environment, the strong attention was put to assure high flexibility and easiness of its configuration. Threat and Attack Simulator allows managing virtual subnetworks and creating multiple virtual network nodes (Figure 4). The network nodes, the hosts, are easily configurable and provided with diverse resources. Particularly they are provided with various software i.e. the operating systems and the specialised programs for developing attacker tools and for performing the attacks.

- *Observer Terminal*, which allows monitoring the traffic of Mirrored Information System in order to evaluate the effects caused by the simulated attacks on the system. It tracks all the malicious or anomalous events happening in Mirrored Information System during tests and experiments and stores them in the central database, to which user access is provided via various interfaces. The system is based on the Intrusion Detection Engine.

- *Vulnerabilities and Countermeasures Repository*, storing information about system vulnerabilities and the

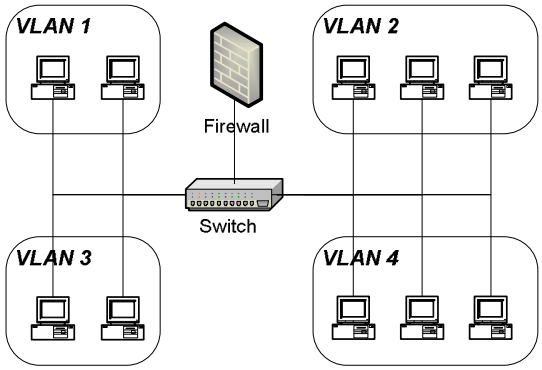

Figure 4: An exemplary setting of Threat and Attack Simulator network.

relative countermeasures. It is composed of two subsystems: the Vulnerabilities and Countermeasures Database and the Binaries Repository. The former stores the knowledge about the existing and known vulnerabilities, threats, attacks and countermeasures, while the latter is devoted to store and catalogue the attack tools, such as packet generators, trojan horses and root-kits, and other executable code to be used in security experiments carried out in the simulation environment. Vulnerabilities and Countermeasures Repository was implemented in the framework of Industrial Security Risks Assessment Workbench (InSAW) [34, 35] which is our proprietary system based on a two tier client/server database driven architecture.

- *Testbed Master Administrator*, used to remotely manage both the network and the experiments. It manages the operations related to initiation and termination of experiments and allows real time observation of each system's behaviour when a simulation is launched.

- *Horizontal Services*, responsible for providing services that are needed for the efficient management of the simulation environment such as backup services or file sharing services.

Further details about the simulation environment can be found in [36].

4. CONCLUSIONS

In the paper we have presented MAlSim – Mobile Agent Malware Simulator, developed to address our needs for simulation tools to be applied during the experiments aiming at evaluation of security threats to critical networked infrastructures.

The framework is based on the technology of mobile agents, which appears to be particularly suitable for this application due to numerous similarities between agents and malicious programs (such as mobility, autonomy etc.) and because of the features of agent platforms which facilitate performance of experiments.

The framework is deployed (through JADE agent platform) in the simulation environment of our cybersecurity laboratory. The environment comprises Mirrored Information System aiming at reconstructing the information system of an evaluated critical networked infrastructure; and supporting sections which, among the others, facilitate configuration, performance and observation of experiments and collection, storage and processing of results.

MAlSim Toolkit provides multiple classes of MAlSim agent and diverse behavioural and migration/replication patterns, to be used for implementation of various malware. At its current state, the repository of agent classes and behaviours contains just basic malware implementations for zero-day viruses and worms, which we were applying during the recent studies on computer security of a power plant. However, in the foreseeable future we are going to extend the repository, providing agent classes and behaviours of other malicious programs.

We are also going to develop agent behaviours aiming at minimising the difference between the simulated conditions and the conditions of simulation, which stems from the fact that MAlSim uses default JADE communication mechanisms realised over Java Remote Method Invocation protocol.

5. REFERENCES

[1] Federal Office for Information Security (BSI). BSI annual report 2003. Internet, 2003. Available at `http://www.bsi.bund.de/english/publications/annualreport/index.htm` (last access: October 30, 2007).

[2] Janusz Górski, Aleksander Jarzębowicz, Rafał Leszczyna, Jakub Miler, and Marcin Olszewski. Trust case: justifying trust in an it solution. In *Proceedings of Safecomp Conference, Reliability Engineering and System Safety*, volume 89, pages 33–47. Elsevier, July 2005.

[3] Ed Skoudis and Lenny Zeltser. *Malware: Fighting Malicious Code*. Prentice Hall Professional Technical Reference, Upper Saddle River, New Jersey, USA, November 2003.

[4] Sarah Gordon. Are good virus simulators still a bad idea? *Network Security*, 1996(9):7–13, September 1996.

[5] Joe Hirst. Virus simulation suite. Internet, 1990.

[6] Thomas Faistenhammer, Martin Klöck, Karlhorst Klotz, Thomas Krüger, Peter Reinisch, and Jenny Wagner. Virlab 2.1. Internet, October 1993. Available at `http://kklotz.de/html/virlab.html` (last access: October 29, 2007).

[7] Rosenthal Engineering. Rosenthal virus simulator. Internet, 1997.

[8] Monirul I. Sharif, George F. Riley, and Wenke Lee. Comparative study between analytical models and packet-level worm simulations. In *PADS '05: Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation*, pages 88–98, Washington, DC, USA, 2005. IEEE Computer Society.

[9] Symantec Research Labs. Symantec worm simulator. Internet, 2005.

[10] Dan Ellis. Worm anatomy and model. In *WORM '03: Proceedings of the 2003 ACM workshop on Rapid malcode*, pages 42–50, New York, NY, USA, 2003. ACM.

[11] Cliff Changchun Zou, Weibo Gong, and Don Towsley. Worm propagation modeling and analysis under dynamic quarantine defense. In *WORM '03: Proceedings of the 2003 ACM workshop on Rapid*

malcode, pages 51–60, New York, NY, USA, 2003. ACM.

[12] Michael Liljenstam, Yougu Yuan, BJ Premore, and David Nicol. A mixed abstraction level simulation model of large-scale internet worm infestations. In *MASCOTS '02: Proceedings of the 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'02)*, page 109, Washington, DC, USA, 2002. IEEE Computer Society.

[13] Michael Liljenstam, David M. Nicol, Vincent H. Berk, and Robert S. Gray. Simulating realistic network worm traffic for worm warning system design and testing. In *WORM '03: Proceedings of the 2003 ACM workshop on Rapid malcode*, pages 24–33, 2003.

[14] Arno Wagner, Thomas Dübendorfer, Bernhard Plattner, and Roman Hiestand. Experiences with worm propagation simulations. In *WORM '03: Proceedings of the 2003 ACM workshop on Rapid malcode*, pages 34–41, New York, NY, USA, 2003. ACM.

[15] David Moore, Colleen Shannon, Geoffrey M. Voelker, and Stefan Savage. Internet quarantine: Requirements for containing self-propagating code. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE*, volume 3, pages 1901–1910, April 2003.

[16] Kalyan S. Perumalla and Srikanth Sundaragopalan. High-fidelity modeling of computer network worms. *acsac*, 00:126–135, 2004.

[17] Songjie Wei, Jelena Mirkovic, and Martin Swany. Distributed worm simulation with a realistic internet model. In *PADS '05: Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation*, pages 71–79, Washington, DC, USA, 2005. IEEE Computer Society.

[18] Songjie Wei and Jelena Mirkovic. A realistic simulation of internet-scale events. In *Valuetools '06: Proceedings of the 1st international conference on Performance evaluation methodolgies and tools*, page 28, New York, NY, USA, 2006. ACM Press.

[19] Mischel Internet Security. Trojan simulator. Internet, 2003. Available at `http://www.misec.net/trojansimulator/` (last access: October 29, 2007).

[20] Fabio Bellifemine, Giovanni Caire, Tiziana Trucco, and Giovanni Rimassa. *JADE - A White Paper*, September 2003.

[21] David Chess, Colin Harrison, and Aaron Kershenbaum. Mobile agents: Are they a good idea? Technical Report RC 19887 (December 21, 1994 - Declassified March 16, 1995), IBM Research, Yorktown Heights, New York, 1994. Available at `citeseer.ist.psu.edu/chess95mobile.html`.

[22] Davis Chess, Benjamin Grosof, Colin Harrison, David Levine, Colin Parris, and Gene Tsudik. Itinerant agents for mobile computing. *IEEE Personal Communications*, 2(5):34–49, 1995. Available at `citeseer.ist.psu.edu/article/chess95itinerant.html`.

[23] S. Franklin and A. Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Intelligent Agents III. Agent Theories, Architectures and Languages (ATAL'96)*, volume 1193, Berlin, Germany, 1996. Springer-Verlag New York, Inc. Available at `citeseer.ist.psu.edu/franklin96is.html`.

[24] Antonio Carzaniga, Gian Pietro Picco, and Giovanni Vigna. Designing distributed applications with a mobile code paradigm. In *Proceedings of the 19th International Conference on Software Engineering*, Boston, MA, USA, 1997. Available at `citeseer.ist.psu.edu/carzaniga97designing.html`.

[25] Alfonso Fuggetta, Gian Pietro Picco, and Giovanni Vigna. Understanding code mobility. *IEEE Transactions on Software Engineering*, 24(5):342–361, 1998. Available at `citeseer.ist.psu.edu/fuggetta98understanding.html`.

[26] Dejan S. Milojicic. Trend wars: Mobile agent applications. *IEEE Concurrency*, 7(3):80–90, 1999. Available at `http://dlib.computer.org/pd/books/pd1999/pdf/p3080.pdf`.

[27] Bennet S. Yee. A sanctuary for mobile agents. In *Proceedings of the DARPA Workshop on Foundations for Secure Mobile Code*, Monterey, USA, March 1997. Available at `citeseer.ist.psu.edu/article/yee97sanctuary.html` (last access: May 08, 2006).

[28] Robert S. Gray, David Kotz, George Cybenko, and Daniela Rus. Mobile agents: Motivations and state-of-the-art systems. Technical Report TR2000-365, Dartmouth College, Hanover, NH, 2000. Available at `citeseer.ist.psu.edu/gray00mobile.html`.

[29] W. Jansen and T. Karygiannis. Nist special publication 800-19 - mobile agent security, 2000. Available at `citeseer.ist.psu.edu/jansen00nist.html`.

[30] Telecom Italia Lab. Java Agent DEvelopment Framework. Website. `http://jade.tilab.com/`.

[31] Giovanni Caire. *JADE tutorial: application-defined content languages and ontologies*, June 2002.

[32] Rafał Leszczyna. Evaluation of agent platforms. Technical report, European Commission, Joint Research Centre, Institute for the Protection and security of the Citizen, Ispra, Italy, June 2004.

[33] Fabio Bellifemine, Giovanni Caire, Tiziana Trucco, and Giovanni Rimassa. *Jade programmerŠs guide*, February 2003.

[34] Marcelo Masera and Igor Nai Fovino. A service oriented approach to the assessment of infrastructure security. In *First Annual IFIP Working Group 11.10 International Conference on Critical Infrastructure Protection*, volume 1, March 2007.

[35] Marcelo Masera Igor Nai Fovino and Alessio Decian. Integration of cyber-attack within fault trees. In *17th European Safety and Reliability Conference (ESREL)*, volume 3, pages 2571–2578, June 2007.

[36] Rafał Leszczyna, Igor Nai Fovino, and Marcelo Masera. Security evaluation of it systems underlying critical networked infrastructures. Accepted for First International IEEE Conference on Information Technology (IT 2008), Gdansk, Poland, 18 – 21 May 2008, May 2008.