



MAISim - Mobile Agent Malware Simulator

Rafał Leszczyna, Igor Nai Fovino, Marcelo Masera



EUR 23026 EN - 2007

The mission of the JRC is to provide customer-driven scientific and technical support for the conception, development, implementation and monitoring of EU policies. As a service of the European Commission, the JRC functions as a reference centre of science and technology for the Union. Close to the policy-making process, it serves the common interest of the Member States, while being independent of special interests, whether private or national.

European Commission
Joint Research Centre

Contact information

Address: Via E. Fermi, 1 – 21027 Ispra (VA) - Italy
E-mail: rafal.leszczyna@jrc.it
Tel.: +39.0332.786715
Fax: +39.0332.789576

<http://www.jrc.ec.europa.eu>

Legal Notice

Neither the European Commission nor any person acting on behalf of the Commission is responsible for the use which might be made of this publication.

A great deal of additional information on the European Union is available on the Internet. It can be accessed through the Europa server
<http://europa.eu/>

JRC 41459

EUR 23026 EN

ISSN 1018-5593

Luxembourg: Office for Official Publications of the European Communities

© European Communities, 2007

Reproduction is authorised provided the source is acknowledged

Printed in Italy

Contents

Introduction	5
1 Related work	7
2 Simulation Environment	9
2.1 MAISim Framework	10
2.2 Simulation Lab Environment	14
2.3 Threat and Attack Simulator	14
2.4 Observer Terminal	15
2.5 Vulnerabilities and Countermeasures Repository	17
2.6 Testbed Master Administrator	20
2.7 Horizontal Services	20
2.8 Case Study: Simulation of Zero-Day Virus Attack	20
3 Conclusions	23

Introduction

This report introduces MAISim – Mobile Agent Malware Simulator, a mobile agent framework developed to address one of the most important problems related to the simulation of attacks against critical infrastructures¹ i.e. 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.

Our Action: Security of Critical Networked Infrastructures (SCNI) aims at facilitating the description, assessment and governance from the security point of view of critical networked infrastructures, including information systems, communication networks, electricity and other energy networks and water networks. The main interest is in cross-border and European-wide issues.

The action concentrates on the cybersecurity and topological aspects of infrastructures and their interdependencies, and studies their vulnerabilities (at the technological and system levels), 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. It also studies the conditions and potential means for making decisions on security matters, estimating the impact of these decision, and facilitating the interaction among the stakeholders.

The focus is on providing policy makers and the stakeholders of critical infrastructures with information and 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. From the technological perspective, the action studies the security of industrial control systems (e.g. SCADA, protection and defence systems, monitoring systems), of communication infrastructures (e.g. Internet protocols and WAN), and their application in concrete industrial environments (e.g. electric power).

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 observa-

¹Critical 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 [10]. Today most of critical infrastructures depend highly on the underlying communication networks.

tions 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 constitute 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.

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 [29].

Malware can be categorised into the following families [29]:

- 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.
- 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 (MAISim) was created. MAISim is a software framework which aims at simulation of various malicious software in computer network of an arbitrary information system.

The report is organised as follows: in Chapter 1 we present a brief overview of the related works. In Chapter 2 we describe MAISim and the simulation environment, in which the framework is deployed. Finally, in Chapter 3 we present our conclusions.

Chapter 1

Related work

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 [13]. 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' [15]. Another example is Virlab [9] 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 seems that only one solution of this type was developed [13], namely Rosenthal Virus Simulator [27]. 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' [27]. 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 [8,28,30,36], which base on epidemiological equations that describe spread of real-world diseases. The empirical approaches concentrated mainly on single-node worm spread simulators [19,20,24,32], which are dedicated to run on one machine. Only few distributed worm simulations were implemented [25,33,34] 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.

Also Trojan Simulator [23] 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.

Chapter 2

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 2.1). 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.

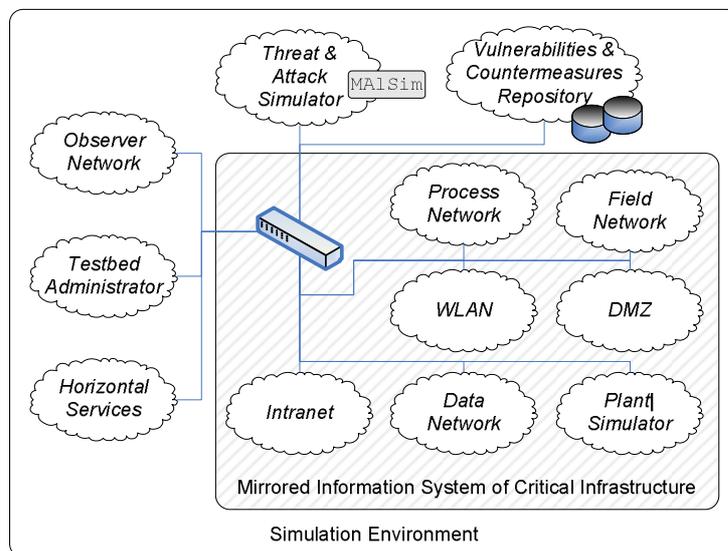


Figure 2.1: Simulation environment.

2.1 MAISim Framework

MAISim – (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).

MAISim Framework was developed using the technology of *mobile agents*. *Software agents* are software components, that are [2]:

- *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 MAISim 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 another¹.

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

JADE is a fully Java based agent platform which complies with the FIPA² 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 [4, 31].

Further details on the choice of JADE for our works can be found in [18].

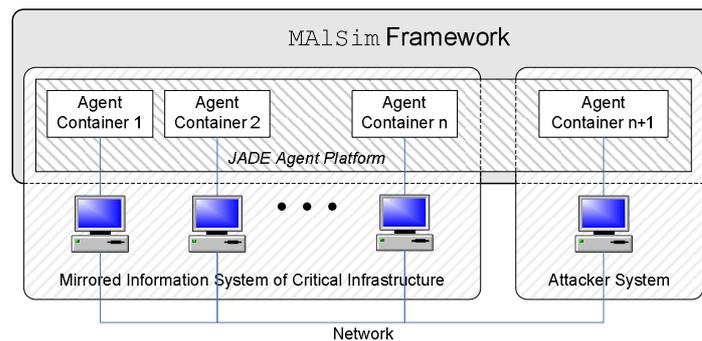


Figure 2.2: MAISim deployment.

As shown in Figure 2.2 JADE has to be deployed over all hosts participating in the experiments with MAISim. 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.

MAISim Toolkit provides:

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

¹Further information on software agents an interested reader can find in [5–7, 11, 12, 14, 17, 22, 35].

²www.fipa.org

- Various behavioural patterns implemented as agent behaviours³ (extensions of Behaviour class).
- Diverse migration/replication patterns implemented as agent behaviours (extensions of Behaviour class).

The MAISim 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.

Migration and replication patterns describe the ways in which MAISim 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 the introduction, 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). MAISim on the other hand uses the migration mechanisms embedded in the agent platform.

In the default configuration (used for the MAISim 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

³In 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 [3].

of simulation. Thus we are going to develop agent behaviours aiming at minimising this difference. One solution could be for example to not allow MAISim agent migrate until a transport channel used by the prototype malware was opened. As a result, MAISim agent, even if 'physically' moving through the connection on 1099 port, will behave as relocating through a HTTP or POP3 connection etc.

During MAISim 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 MAISim agent from the database and configure it. If none of existing MAISim agents fit the attack scenario, we develop a new MAISim agent.
3. We extend the agent with migration schema (through adding agent behaviours from the repository).
4. We extend the agent with malicious behaviour.

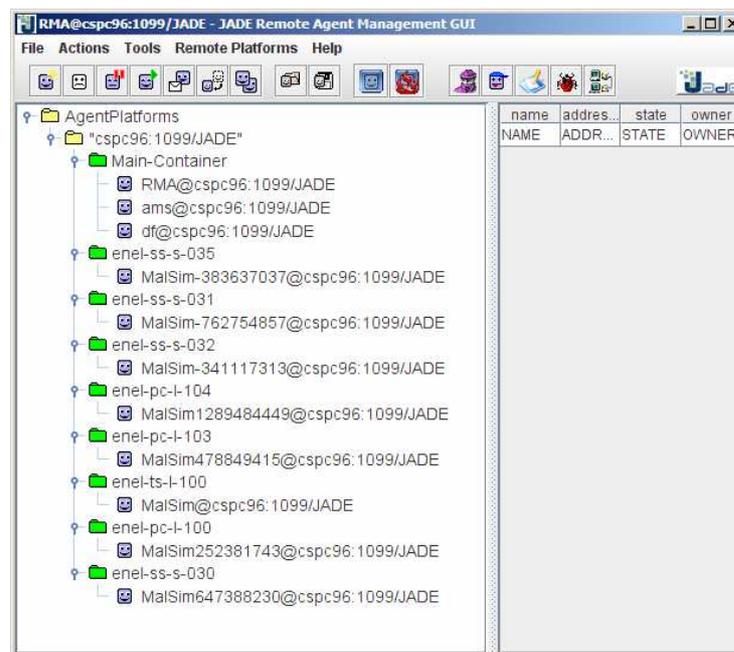


Figure 2.3: MAISim Framework takes advantage of JADE GUI for control and observation of experiments.

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 2.3 the graphical console allows also observation of the diffusion of the simulated malware.

2.2 Simulation Lab Environment

Malware propagation features depend heavily on the characteristics of the underlying network [34]. 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 production process), the field network (interconnecting controllers and field devices), the corporate network etc. (see Figure 2.1).

However it must be noted that MAISim 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 jeopardize the analysed information system.
- *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.
- *Vulnerabilities and Countermeasures Repository*, storing information about system vulnerabilities and the relative countermeasures.
- *Testbed Master Administrator*, used to remotely manage both the network and the experiments.
- *Horizontal Services*, which provides support services such as FTP server, backup service etc. to other subsystems.

2.3 Threat and Attack Simulator

Threat and Attack Simulator is the part of simulation environment where the simulated attacks are configured and launched.

There is a great number of already identified and documented attacks and there are also many attacks yet unknown to the computer security community. All of the attacks have diverse characteristics and they are performed in different ways. Some of them originate on one host, but usually they require a chain of hosts (to amplify the strength of attack or to make the attacker untraceable etc.). The attacks require diverse attacker skills, from very basic (running an automated application) to very advanced (extensional developing skills, deep knowledge of computer system

vulnerabilities etc.). Furthermore to perform the attacks diverse types of applications and operating systems must be used. In general, attacks of different types can strongly differ in hardware (including attacker network topology) and software resources required to perform them.

Thus one of the primary characteristics of a system aiming at simulating attacks is its flexibility and easiness of configuration. The crucial feature of Threat and Attack Simulator is its flexibility in managing virtual subnetworks and creating multiple virtual network nodes. 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 (various Linux distributions, Microsoft Windows versions etc.), and the specialised programs for developing attacker tools and for performing the attacks.

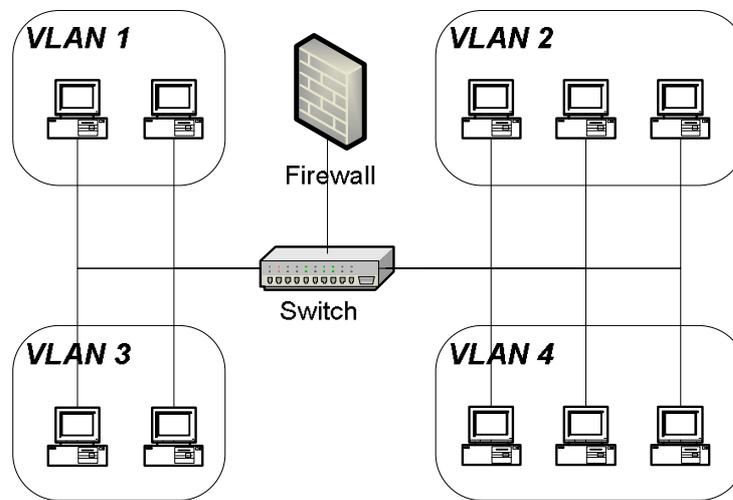


Figure 2.4: An exemplary setting of Threat and Attack Simulator network.

The topology of emulated fragments of Internet networks, as well as hosts which form a part of this topology, and their resources, change depending on needs. And the needs are determined by each particular attack. The subsystem is reconfigured depending on the simulated attack. The system topology is defined and designed in reference to each particular attack which is to be performed in its boundaries. An exemplary network topology of Threat and Attack Simulator with four VLAN networks is presented in Figure 2.4.

2.4 Observer Terminal

The scope of the Observer system is to keep track of all the malicious or anomalous events happening in Mirrored Information System during tests and experiments.

The system is based on the Intrusion Detection Engine. The implemented architecture is composed of the following elements:

- Sensors, which capture the network traffic, analyse it and identify the traffic patterns which satisfy some predefined criteria.
- Observer Network, which delivers the packets selected by the sensors to a central collector.
- Observer Database, collecting data received by different sensors distributed in the whole system, and providing support for database querying.
- User Consoles, providing interfaces for access of data collected in the database.

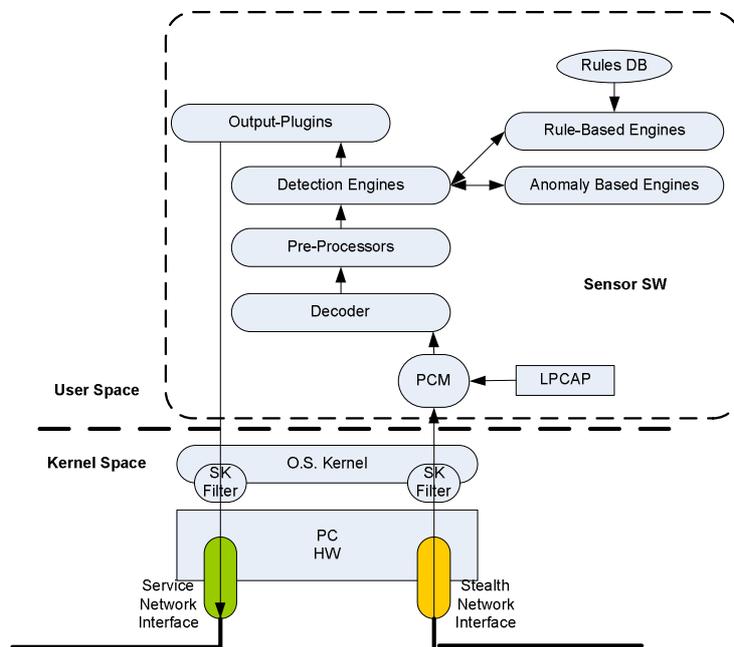


Figure 2.5: Observer Sensor structure.

For the implementation of the sensors we have adopted the Snort Technology [26]. The technology supports the sensor operation in four different modes:

- Sniffer mode – packets are captured and displayed directly on a console.
- Packet Logger mode – packets are logged to disk.
- Network Intrusion Detection System mode – the most complex and configurable setting, which supports analyses of network traffic based on matches against a user-defined sets of rules.

- Inline mode – packets are collected from iptables instead of libpcap allowing control over packet flow.

In our system, the sensors are set to the Network Intrusion Detection System mode. Figure 2.5 presents the structure of an observer sensor.

The sensors track anomalies in the network and send alerts triggered by the discovery of such anomalies to the Observer Database. The alerts are first sent to the Observer Database server called Real-Time Repository. Then, after the experiment termination, they are automatically transferred to the second Observer Database server, namely the Archive Repository. This configuration allows keeping separated the data of past experiments and the data of the experiment in progress, which facilitates the analysis phase. Moreover keeping the two databases separated allows access to the data of past experiments without affecting performance of alerts collection of the ongoing experiment.

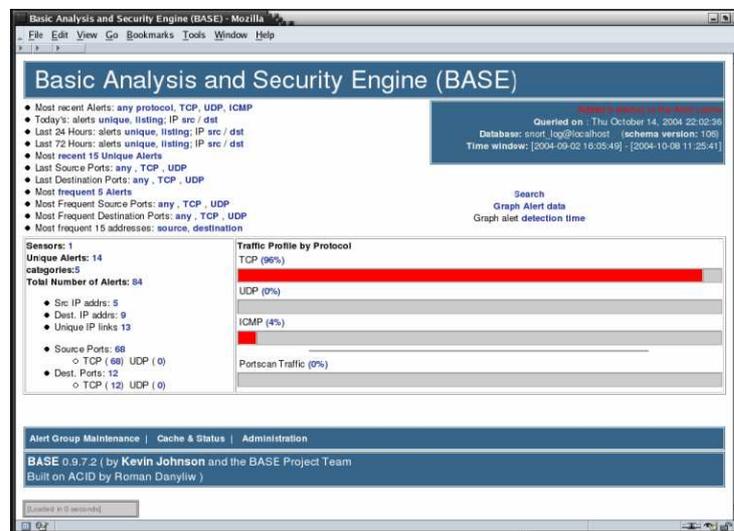


Figure 2.6: The graphical web interface of Observer Database is implemented in BASE.

The graphical web interface for the remote access to Observer Database is implemented in BASE framework (see Figure 2.6). BASE [1] is a web interface to perform analyses of intrusions detected by Snort. It provides authentication and access-control mechanisms.

2.5 Vulnerabilities and Countermeasures Repository

The aim of Vulnerabilities and Countermeasures Repository is to provide a structured knowledge base on vulnerabilities, threats, attacks of information systems and on the possible countermeasures.

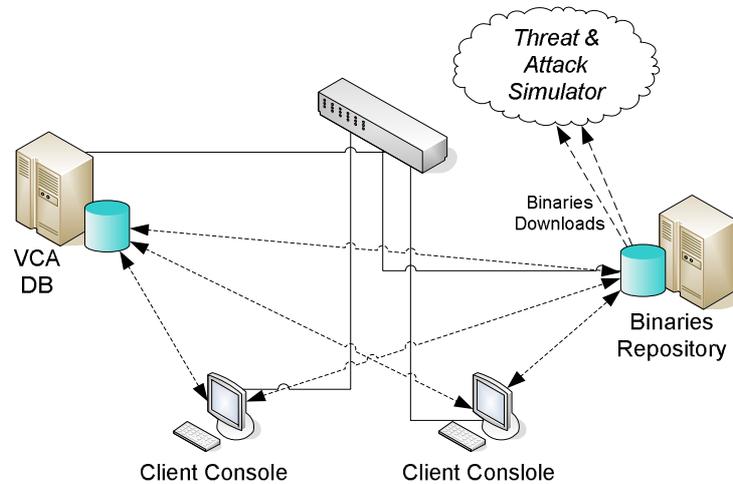


Figure 2.7: Vulnerabilities and Countermeasures Repository. The dashed line arrows represent the data flows among the components of the system.

As it is shown in Figure 2.7 the system is composed of two sub-systems: the Vulnerabilities and Countermeasures Database and the Binaries Repository.

The Vulnerabilities and Countermeasures Database stores the knowledge about the existing and known vulnerabilities, threats, attacks and countermeasures. The Binary Repository 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.

We categorize the following types of information stored in the Vulnerabilities and Countermeasures Repository:

- Off-line documentation – documents of various type (text files, word processor documents, graphical files etc.) stored locally in the file-system of the Vulnerabilities and Countermeasures Database.
- On-line documentation – information about vulnerabilities, attacks and countermeasures located in external web-repositories of the Internet. This information is linked to the database by external references.
- Binary codes – binary codes of patches, attack tools and other executables stored in the Binaries Repository.

Vulnerabilities and Countermeasures Repository was implemented in the framework of Industrial Security Risks Assessment Workbench (InSAW) [16,21] which is our proprietary system based on a two tier client/server database driven architecture. Developed with Microsoft Visual Studio 2005 and Microsoft .Net Framework 2.0. InSAW facilitates assessment of vulnerabilities, threats, attacks and risks through a set of libraries and the graphical interface which allows creation of graphs and charts.

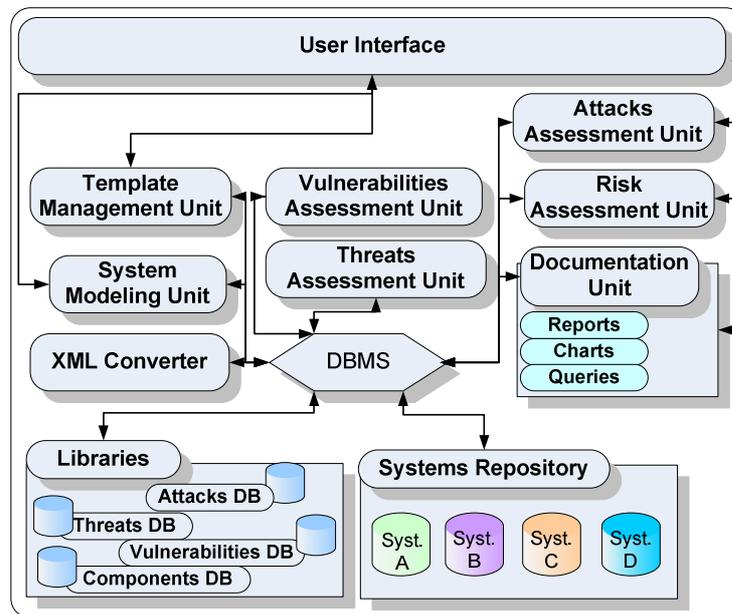


Figure 2.8: Architecture of Vulnerabilities and Countermeasures Repository.

The system is composed of the following modules (Figure 2.8):

- SQLServer-based Database Management System for management of internal data repositories.
- Libraries, which store the proper Vulnerabilities and Countermeasures related information classified into:
 - Vulnerabilities – categorised by vulnerability type, severity etc.
 - Components – the vulnerable system elements.
 - Threats – categorised by target, type and severity.
 - Attacks – in the form of attack trees organised by type.
 - Countermeasures – possible measures to avoid instantiation of threats.
- Systems Repository – a collection of databases, one for each analysed system.
- Analysis Engines – analysis engines for the evaluation of different aspects related to vulnerabilities, attacks, threats and countermeasures.
- Querying, Reporting and Chart Units – facilitating data retrieval and representation (by means of reports, charts and tables).
- System Aided Modelling Unit, which provides the interface for modelling of vulnerabilities, attacks, threats and complex systems.

2.6 Testbed Master Administrator

The Testbed Master Administrator system brings in tools for remote control and monitoring of the hosts participating in 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.

Since the remotely controlled hosts are supplied with various operating systems (Windows and Linux), we employ diverse remote management solutions. For Windows we take advantage of UltraVNC and Windows Remote Desktop. For Linux, a remote management system based on SSH was applied, and the Putty SSH client was installed.

2.7 Horizontal Services

The Horizontal Services system is responsible for providing services that are needed for the efficient management of the simulation environment. At the current stage the two services have been implemented:

- Backup Service, for restoring initial simulation conditions before each experiment and to prevent from loss of data in case of hardware failure or accidental erasure. Backup Service is implemented using Symantec NetBackup software package, providing high-performance backups and restores for a variety of platforms, including Microsoft Windows and Unix-like operating systems.
- FTP Filesharing Service, supplying the simulation environment with a shared storage area which can be easily accessed from machines running diverse operating systems. The service is set up on open source FileZilla FTP Server, which supports FTP and FTP over SSL/TLS, with a secure encrypted connection between client and server. FileZilla also provides on-the-fly data compression significantly improving transfer rates.

2.8 Case Study: Simulation of Zero-Day Virus Attack

As mentioned in Section 2.2 we have applied the framework for the experiments aiming at evaluation of the security of a power plant infrastructure.

In order to perform this evaluation we reconstructed the network setting of the power plant i.e. we mirrored (Figure 2.1):

- The process network, which interconnects diverse subsystems of the energy production process.
- The field network, which interconnects controllers and field devices.
- The corporate network (Intranet).

- Wireless LAN network.
- Demilitarised Zone (DMZ).

We deployed the JADE framework over the hosts mirroring Power Context and Process Network. On each of the hosts we installed a representative JADE container.

In this setting we have performed the simulation of a *zero-day* virus attack. A zero-day (or zero-hour) attack is a computer threat that exposes undisclosed or unpatched computer application vulnerabilities. Zero-day attacks take advantage of computer security holes for which no solution is currently available. Zero-day exploits are released before the vendor patch is released to the public. A zero-day exploit is usually unknown to the public and to the product vendor.

We have developed the attack scenario and based on this scenario we have performed all its steps.

The scenario of the attack is as follows:

A power plant operator working on a PC in Power Context network browses the Internet and gets accidentally infected by a virus which has been just launched in the recent hours. This is a new type of virus, not just a slight modification of an existing one. For this reason and because of the fact that the virus is so recent, it is yet unknown to the antivirus community (zero-day virus). Its signature is not stored in any of antivirus databases. The virus infects programs on the user's PC and, taking advantage of the fact that the traffic between the hosts in the Power Context network is allowed, it infects also the remaining hosts of the Power Context network. Later on the user, unconscious of the fact that his/her PC is infected by the virus, opens the VPN connection to a host in Process Control network. Now the virus has a free passageway to the critical subnetwork of the power plant network. It moves through it and starts infecting the computers in the Process Control network. Simultaneously, the adverse effects of the virus begin to be apparent. The computers become less effective, the applications raise errors and stop functioning, and the network connections are lost.

The general aim of this attack is to infect as many computers in the Internet as possible and to cause their malfunctioning. The attack is not particularly oriented against the power plant system, however when reaching the network of the power plant, the virus can reach the Process Control Network and Power Context subsystems.

In the simulation, the MAISim agent had been launched at main-container and after that it was creating its copies gradually on the hosts in Power Context and progressively in Process Network, starting from SCADA Server. At all of the computers it deactivated the network card making any network-related operation impossible.

The experiment showed:

- The impact of the attack on the operation of the critical infrastructure can be very strong. In worst case the attack can result in the lost of synchronisation

of the traffic between the SCADA servers and the Field actuators, which can lead to malfunctioning of the power plant.

- It is very difficult to prevent from this attack since its strength is based on its urgency and unexpectedness. Most of antimalware software, being signature based, will be not prepared for detection of this attack, and will let the virus spread over the networks.
- A possible solution for protection from this type of attacks it to use anomaly detection based malware detection engines.

Chapter 3

Conclusions

In the report we have presented MAISim – 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.

MAISim Toolkit provides multiple classes of MAISim 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 MAISim uses default JADE communication mechanisms realised over Java Remote Method Invocation protocol. Moreover, in the close future we are going to install the agent simulation platform in alternative cybersecurity laboratory.

Bibliography

- [1] Basic analysis and security engine. Internet, 2003. Available at <http://base.secureideas.net/index.php> (last access: October 30, 2007).
- [2] Fabio Bellifemine, Giovanni Caire, Tiziana Trucco, and Giovanni Rimassa. *JADE - A White Paper*, September 2003.
- [3] Fabio Bellifemine, Giovanni Caire, Tiziana Trucco, and Giovanni Rimassa. *Jade programmer's guide*, February 2003.
- [4] Giovanni Caire. *JADE tutorial: application-defined content languages and ontologies*, June 2002.
- [5] 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.
- [6] 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.
- [7] Davis Chess, Benjamin Grosf, 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.
- [8] Dan Ellis. Worm anatomy and model. In *WORM '03: Proceedings of the 2003 ACM workshop on Rapid malware*, pages 42–50, New York, NY, USA, 2003. ACM.
- [9] 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).
- [10] 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).

- [11] 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.
- [12] 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.
- [13] Sarah Gordon. Are good virus simulators still a bad idea? *Network Security*, 1996(9):7–13, September 1996.
- [14] 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.
- [15] Joe Hirst. Virus simulation suite. Internet, 1990.
- [16] 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.
- [17] W. Jansen and T. Karygiannis. Nist special publication 800-19 - mobile agent security, 2000. Available at citeseer.ist.psu.edu/jansen00nist.html.
- [18] 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.
- [19] 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 malware*, pages 24–33, 2003.
- [20] 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.
- [21] 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.
- [22] 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>.

- [23] Mischel Internet Security. Trojan simulator. Internet, 2003. Available at <http://www.misec.net/trojansimulator/> (last access: October 29, 2007).
- [24] David Moore, Colleen Shannon, Geoffrey M. Voelker, and Stefan Savage. Internet quarantine: Requirements for containing self-propagating code. In *IN-FOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*. IEEE, volume 3, pages 1901–1910, April 2003.
- [25] Kalyan S. Perumalla and Srikanth Sundaragopalan. High-fidelity modeling of computer network worms. *acsac*, 00:126–135, 2004.
- [26] Martin Roesch. Snort – lightweight intrusion detection for networks. Internet, 2003. Available at <http://www.snort.org/docs/lisapaper.txt> (last access: October 30, 2007).
- [27] Rosenthal Engineering. Rosenthal virus simulator. Internet, 1997.
- [28] 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.
- [29] Ed Skoudis and Lenny Zeltser. *Malware: Fighting Malicious Code*. Prentice Hall Professional Technical Reference, Upper Saddle River, New Jersey, USA, November 2003.
- [30] Symantec Research Labs. Symantec worm simulator. Internet, 2005.
- [31] Telecom Italia Lab. Java Agent DEvelopment Framework. Website. <http://jade.tilab.com/>.
- [32] 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 malware*, pages 34–41, New York, NY, USA, 2003. ACM.
- [33] Songjie Wei and Jelena Mirkovic. A realistic simulation of internet-scale events. In *Valuetools '06: Proceedings of the 1st international conference on Performance evaluation methodologies and tools*, page 28, New York, NY, USA, 2006. ACM Press.
- [34] Songjie Wei, Jelena Mirkovic, and Martin Swamy. 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.
- [35] 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).

- [36] 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 malware*, pages 51–60, New York, NY, USA, 2003. ACM.

European Commission

EUR 23026 EN – Joint Research Centre

Title: MAISim - Mobile Agent Malware Simulator

Author(s): Rafał Leszczyna, Igor Nai Fovino, Marcelo Masera

Luxembourg: Office for Official Publications of the European Communities

2007 – 30 pp. – 21.0 x 29.7 cm

EUR – Scientific and Technical Research series – ISSN 1018-5593

Abstract

The report introduces MAISim - Mobile Agent Malware Simulator, a mobile agent framework developed to address one of the most important problems related to the simulation of attacks against critical infrastructures i.e. 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.

The mission of the JRC is to provide customer-driven scientific and technical support for the conception, development, implementation and monitoring of EU policies. As a service of the European Commission, the JRC functions as a reference centre of science and technology for the Union. Close to the policy-making process, it serves the common interest of the Member States, while being independent of special interests, whether private or national.

