Proc. of the Second Intl. Conf. on Advances in Computing, Electronics and Electrical Technology - CEET 2014.
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-034-7 doi: 10.15224/ 978-1-63248-034-7-31

Malware Simulation using JADE

Manish Jain and Dinesh Gopalani

***Abstract*— One of the problems related to the simulation of attacks against computer networks 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 networked infrastructures. To address this issue we developed a Malware Simulation Tool using JADE. The framework uses the technology of mobile agents and it aims at simulation of various types of malicious software. Moreover it can be flexibly deployed over a real computer network system.**

Keywords—malware, simulation, mobile agent, JADE

I. Introduction

In our work we address the problem of security of computer networks. We study the vulnerabilities and the potential malicious threats that might affect them. We reconstruct a computer network in our laboratory and in this configuration we implement attack scenarios.

For a better comprehension of our work, we would like to provide a categorization of the malware first. Malware can be categorized into following families:

- 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. The code for a virus usually consists of a finder, a replicator and often a payload.

- Worms self-replicating programs autonomously (without human interaction) spread across a network. They consume the bandwidth of the network by replicating themselves. The difference from virus is that it does not need any host executable file rather it uses the OS vulnerabilities.

- Trojan horses disguise themselves as useful programs while masking hidden malicious purpose. Trojan horse is actually a non-self-replicating malware that appears to be a legitimate soft- ware and perform a desirable function for the user but instead facilitates unauthorized access to the user's computer system.

- Other Malware include Keystroke Loggers, Botnets, Spyware, Adware, Rootkits etc. Besides, there are also combination of two or more malware.

Our Malware Simulator is a software framework which aims at simulation of various malicious software in a computer network.

The paper is organized as follows: in Section II we present a brief overview of the related works and how our work is different from the those works. In Section III, we describe the methodology used for our Malware Simulator and the simulation environment, in which the framework is deployed. In Section IV, we provide the details of the analysis and the results. Finally, in Section V we present the conclusion. At last we give a list of the references used.

II. 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. 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. Another example is Virlab from 1993, which simulates the spread of DOS computer viruses. 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.

- Concerning the simulation of worms, the prevalent work was done on developing mathematical models of worm propagation, which are based on epidemiological equations that de- scribe spread of real-world diseases. The empirical approaches concentrated mainly on single- node worm spread simulators, which are dedicated to run on one machine. Only few distributed worm simulations were implemented but they

Manish Jain
Computer Science Department, MNIT, Jaipur
India

Dinesh Gopalani
Computer Science Department, MNIT, Jaipur
India

Proc. of the Second Intl. Conf. on Advances in Computing, Electronics and Electrical Technology - CEET 2014.
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-034-7 doi: 10.15224/ 978-1-63248-034-7-31

approach modeling of worm propagation in the Internet and thus they don't run arbitrary network of predefined topology.

In contrast to the alternative works on malware simulation which use modelling approaches to reconstruct the underlying network, we build a real, physical computer network in our laboratory.

III. Simulation Environment

Our Mobile Agent Malware Simulator Framework can be thought of as a software toolkit which aims at simulation of various malicious software in a computer network. 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.).

To understand the concept of mobile agent, we would like to provide a brief description of software agents here.

Software agents: The simplest definition that could be given for software agents is that these are software that have their own IQ and hence can act on user's behalf. Typically, the concept of an agent is described as a complex software entity that is a self-contained object capable of acting with a certain degree of autonomy in order to accomplish tasks on behalf of its user in an intelligent manner and that is responsive to changes in the environment without requiring constant human guidance or intervention. Ideally, an agent makes assumptions based on preferences you've defined, or that it has learned by analysing your behaviours.
Software agents are software programs that are:

- Autonomous - have control over their own actions
- Social - able to interact with other agents
- Pro-active - take initiative without human intervention and respond to change in environment

Simplest example of software agent could be the Animated Help Agent in Microsoft Word which learns from user's previous interactions and then recommend actions accordingly.
A Mobile Agent, namely, is a type of software agent, with the additional feature of mobility. More specifically, a mobile agent is a process that can transport its state from one environment to another, with its data intact, and be capable of performing appropriately in the new environment. When a mobile agent decides to move, it saves its own state, trans- ports this saved state to the new host, and resumes execution from the saved state.
Figure 1 shows the use of Mobile Agent for the processing of data at three different locations without the transfer of the data to be processed. Instead of data going to the process, it is now that the process itself goes to the location of the data which saves a lot of network bandwidth.
We chose the concept of Mobile Agents for my project for the following reasons:

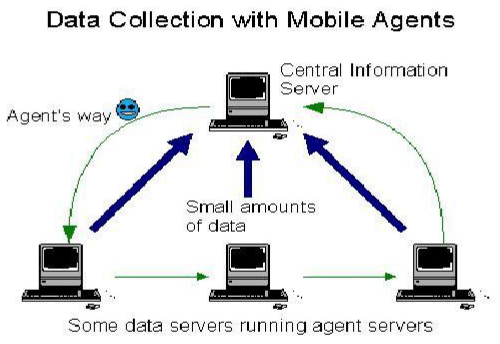
Data Collection with Mobile Agents
Central Information Server
Agent's way
Small amounts of data
Some data servers running agent servers

Figure 1. Mobile Agents

Mobile agents are software agents that are able to roam network freely and to spontaneously re- locate themselves from one device to another.

- Mobile agents have much in common with malicious programs.
- Similar to worms and viruses, they have the ability of relocating themselves from one computer to another.
- They are also autonomous as the worms are.

Hence we used mobile agents and their properties for the purpose of transferring and executing malicious code from one system to another in our simulation environment.

We used JADE (Java Agent Development Frame- work) which a multi agent platform based on Java which compiles with the The Foundation for Intelligent Physical Agents (FIPA) specifications.

Java Agent Development Framework, or JADE, is a software framework for multi-agent systems, in Java that has been in development since at least 2001. The JADE platform allows the coordination of multiple FIPA-compliant agents and the use of the standard FIPA-ACL communication language.

JADE is completely implemented in Java language and the minimal system requirement is the version 1.4 of JAVA (the run time environment or the JDK). JADE is free The latest version of JADE is JADE 4.0 released on 20/04/2010 distributed by Telecom Italia, the copyright holder, in open source software under the terms of the LGPL (Lesser General Public License Version 2).

JADE proved to be a good choice for our Malware Simulation project because:

Proc. of the Second Intl. Conf. on Advances in Computing, Electronics and Electrical Technology - CEET 2014.
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-034-7 doi: 10.15224/ 978-1-63248-034-7-31

- it had all the agent features that we needed (and more)
- communication between mobile agents running on various workstations on the network was trivial to do
- it is efficient and tolerant of faulty programming
- it follows FIPA standards
- the user group is very active and implementors typically respond to problems within 24 hours

After downloading and extracting the Jade folders onto a computer in the network, it is necessary to install and run "ANT". Ant is a build tool which runs on XML files in order to compile the java source files (present in JADE) and creates a desired deployment structure. Ant version 1.8.0 is freely downloadable from: http://ant.apache.org/

For running ANT, it is necessary to set the three environment variables, namely ANTHOME: ant's di- rectory, PATH: ant's bin directory, JAVAHOME: jdk directory of Java.

When we run ant, it acts in accordance with the build.xml file present in the Jade directory and com- piles all the source files to create corresponding class files. Ant creates the desired deployment structure of all the class files as specified in the build.xml file by putting them into respective output directories. After running ant, we can launch the container and agents therein.

For launching Jade, first of all we set the CLASS- PATH to include the JAR files in the lib subdirectory and the current directory.

Now in order to launch Jade on the main container we use the following command:

```
java jade.Boot -gui
```

And similarly to launch Jade on another host in the network we use the following command:

```
java jade.Boot -host 8.1.3.1  -container
```
where

8.1.3.1 is the IP of the system on which the main container is launched.

Each platform must have a Main Container which holds three specialized agents called the AMS agent, the DF agent and the RMA agent.

In a nutshell, Using Jade we run various agents on different containers. Containers are the host computers in the network on which Jade is running. There is one Main Container and then all the other terminals in the network connect to this Main Container. Communications between containers in the same platform (i.e. attached to the same Main Container) in JADE uses RMI and requires full connectivity i.e. each container must be able to open a socket to all other containers in the platform.

Once JADE has been successfully installed and launched on all the systems of the experiment net- work, we launch mobile agent on the main container using the MobileAgent class provided with Jade. After launching mobile agent on the main container, it can be easily moved onto any of the other containers by using the MOVE option provided in the mobile agent GUI. When we move a mobile agent to a new container, the GUI disappears from the main container and is showed off at the desired container.

We used this mobility feature of the mobile agent in JADE to transfer and affect malware from the main container to any other container on the network. Thus in a way, the main container acts as the attacker system and all other computers on the network are its victims.

At the boot time of JADE, I transfer one batch file using RMI from the main container to the other container which

a. acts as a RMI file client

b. gets a malware file from a specified location of the main container to the container on the MOVE event of mobile agent

c. Executes this malware file at the container

The batch file gets executed at the MOVE event of the Mobile Agent and when the mobile agent moves to any of the desired victim container, this file acts as a RMI FileClient and fetches a virus called "act- movie.exe" from a specified location of the Main Container. Next this virus is made to move to the system32 folder of the victim computer and then at last executed to cause the harm.

In order to transfer and run the batch file when the mobile agent moves, we extended the MobileAgent class present in the classes/examples/mobile folder of the JADE directory to write our own MobileAgent class.

We analyzed the MobileAgent class and found that an "aftermove" function runs when a mobile agent is transferred from the main container to another container. We override this "aftermove" function while extending the MobileAgent class to execute the batch file at the targeted container. Thus, when the mobile agent migrates from the Main Container (attacker) to any other (victim) container, the malicious program gets executed and the victim container gets compromised.

After importing the java.lang.* and java.io.* packages, the runtime class is used to run a batch file.

Besides, we also wrote few virus codes directly into the MobileAgent.java file (in Java) at the "aftermove" function which gets executed automatically when the mobile agent is migrated. These Java viruses have the advantage of not getting detected by any of the antivirus softwares as the only process they commence is Java.exe which does not have any malicious signature.

IV. Analysis And Results

In order to affect the computers on the network, we experimented with n-number of malware programs which includes various viruses, trozans, scripts etc.

Proc. of the Second Intl. Conf. on Advances in Computing, Electronics and Electrical Technology - CEET 2014.
Copyright © Institute of Research Engineers and Doctors, USA .All rights reserved.
ISBN: 978-1-63248-034-7 doi: 10.15224/ 978-1-63248-034-7-31

For this purpose, we created a lot of viruses in VBScript, Java and few using Virus Generation Pro- grams.
Following gives a brief list of the viruses used in our project with little description of each:

- CDROM Eject.vbs - This virus created using VBScript opens the CD-ROMS (as many as are there) of the compromised computer once it gets executed there. And the worst part is that it does not even let the user close the CDROM. If the user closes the CDROM, it gets opened again. This was done using the "oWMP.cdromCollection" class of VBScript and then using the Eject function therein within an infinite for loop.

- Notepad.bat - This simple virus infinitely opens Notepad on the compromised computer. This was done using goto label in the batch file which makes the execution process jump back to where the Notepad.exe is being called. This exhausts the processor of the compromised computer and finally the computer hangs.

- ColoredCommandPrompt.bat - This virus opens up a command prompt on the compromised computer and starts drawing colourful lines over it. This covers up the complete screen and devoids the user from accessing even the Desktop. Ctrl+Alt+Delete is the only choice to get rid off.

- Screenshot.java - This virus, written in Java directly into the aftermove function of the mobile agent file, takes a screenshot of the desktop screen of the compromised computer and pre- pares a jpeg file of the same. This way important information can be stolen from the Desktop of any remote PC. The virus as written in Java cannot be detected using any antivirus. In this code, we used the createScreenCapture function provided in the robot class in Java.

- Shutdown.java - This simple virus shuts down the compromised computer without any choice left for the user, not even the time to save any unsaved work. It simply makes use of the run- time class of java to execute the shutdown command at the remote computer. As the virus code is written in the mobile agent file in Java, again it gets tough to detect it.

- Corrupt.java - This java virus when acts at the MOVE of the mobile agent, searches for all the .class files on the victim computer and corrupts them by adding unnecessary into them. Dangerous for java programmers to handle.

- CDROMEject.java - This one is similar to the CDROMEJECT.vbs virus except that it is written in Java and hence becomes difficult to be detected.

- Consume.java - This java virus consumes the complete processor and makes the victim computer handicapped with no space left for other running applications. At slower processors, even the Task Manager cannot be initiated to kill the Java.exe process to stop its effect.

In the code, we create one thread using the Thread class in Java and then set its priority to (Thread.MAXPRIORITY) using the setPriority function in the Thread class. Then we run this thread into an infinite while loop thereby consuming the complete processing capability of the compromised computer.

- Shutdown.java - This one is also similar to Shut- down.bat except for the fact that it was written in Java using the Runtime class.

- CrazyMouse.exe - This virus was created using a virus generation program JPS Virus Maker. It is a .exe virus which makes the mouse of the infected computer bubbling like Crazy and hence the user looses complete control over the mouse pointer.

- DisableCommandPrompt.exe - This virus disables the command prompt of the infected computer until next reboot. It was also created using the JPS Virus Maker.

- DisableControlPanel.exe - This virus disables the Control Panel of the infected computer leaving the user surprised. The only solution left is to restart the computer. It was also created using the JPS Virus Maker.

- DisableStartButton.exe - As obvious from the name, this virus disables the start button from the Taskbar and hence devoids the user from using the Start Menu and Programs. It was also created using the JPS Virus Maker.

- SwapMouseButtons.exe - This annoying virus swaps the function of the two mouse clicks. The right click acts as the left click and the left click acts as the right click. It was also created using the JPS Virus Maker.

We collected a total of 6000 viruses from different sources over the internet and tried to simulate their malicious affects using the mobile agents. Besides, we also tried to analyse the harmful affects of few trojans for e.g. Mellisa as a effect of which computer starts producing infected documents. As a part of experiment we also downloaded API Monitor tool from www.rohitab.com/apimonitor.

ISBN: 978-1-63248-034-7 doi: 10.15224/ 978-1-63248-034-7-31

API Monitor is a free tool which can be used to monitor and display API calls made by applications. We used this tool for observing the System Calls made to DLLs during the execution of our malicious pro- grams.

Thus as a result of all this exercise, we observed the actions performed by various malicious software such as scanning for vulnerabilities of operating system or verifying if certain conditions are met and then executing themselves for some destructive purpose.

V. Conclusions

In the paper we have presented Mobile Agent Malware Simulator using JADE, developed to address our need for simulation process to be applied during the experiments aiming at evaluation of security threats to computer networks.

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

Our system provides multiple classes of mobile 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 viruses and worms. However, in the foreseeable future we are going to extend the repository, providing agent classes and behaviours of other malicious programs. Also in near future, we plan to extensively use the API Monitor Tool in order to prepare a comparison table of the various API Calls made by various malicious and benign programs. Usage of SNORT as an intrusion detection engine is also a planned activity.

During the work we also noticed some compatibility issues among the various operating systems, such as the Windows XP container was not compatible with the container created on Windows 7,while using the Remote Method Invocation protocol.

References

[1] R. Leszczyna, I. N. Fovino, and M. Masera, "MAlSim–Mobile Agent Malware Simulator," in Proc. of 1st International Conference on Simulation Tools and Techniques, Marseille, France, March 03-07, 2008.

[2] A. Wagner, T. D. Aubendorfer, B. Plattner, and Roman Hiestand, "Experiences with worm propagation simulations," in Proc. of 2003 ACM workshop on Rapidmalcode, pp. 34-41, New York, USA, 2003.

[3] D. Chess, C. Harrison, and A. Kershenbaum, " Mobile agents: Are they a good idea?," Technical Report RC 19887, IBM Research, Yorktown Heights, New York, 1994. Available at citeseer.ist.psu.edu/chess95mobile.html.

[4] S. Franklin, and A. Graesser, "Is it an agent, or just a program?: A taxonomy for autonomous agents," in Proc. of Workshop on Intelligent Agents III, Agent Theories, Architectures, and Languages, pp. 21-35, Berlin, Germany, 1996. Available at citeseer.ist.psu.edu/franklin96is.html.

[5] A. Fuggetta, G. P. Picco, and G. Vigna, "Understanding code mobility," IEEE Transactions on Software Engineering, vol. 24(5), pp. 342 - 361, 1998. Available at citeseer.ist.psu.edu/ fuggetta98understanding.html.

[6] D. S. Milojicic, "Trend wars: Mobile agent applications," IEEE Concurrency, vol. 7(3), pp. 80 - 90, 1999. Available at http://dlib.computer.org/pd/books/ pd1999/pdf/p3080.pdf.

[7] R. 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.

[8] S. Gordon, "Are good virus simulators still a bad idea?," Network Security, vol. 1996(9), pp. 7-13, 1996.

[9] F. Bellifemine, G. Caire, T. Trucco, and G. Rimassa, "Jade programmerSs guide," February 2003.

[10] B. S. Yee, " A sanctuary for mobile agents," in Proc. of DARPA Workshop on Foundations for Secure Mobile Code, Monterey, USA, March 1997.

[11] R. 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.