A Worm misuse pattern

Eduardo B. Fernandez[1], Nobukazu Yoshioka[2], and Hironori Washizaki[3]

1 Dept. of Comp. Science and Eng., Florida Atlantic University, Boca Raton, FL, USA, ed@cse.fau.edu
2 National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan, nobukazu@nii.ac.jp
3 Waseda University / GRACE Center, National Institute of Informatics, 3-4-1, Okubo, Shinjuku-ku, Tokyo, Japan, washizaki@waseda.jp

Abstract

We have proposed a new type of pattern, the *misuse pattern*. This pattern describes, from the point of view of the attacker, how a type of attack or misuse is performed (what system units it uses and how), provides ways of stopping the attack by enumerating possible security patterns that can be applied for this purpose, and helps analyzing the attack once it has happened by indicating where can we find forensics data as well as what type of data. A catalog of misuse patterns is needed to let designers evaluate their designs with respect to possible threats. We present here a misuse pattern for a generic worm, which describes the essential and typical characteristics of this type of malware. We consider how to stop this malware and we also discuss some examples and variations.

Introduction

In order to design a secure system, we first need to understand the possible threats to the system. Without this understanding we may produce a system that is more expensive than necessary, it is hard to administer, and has a large performance overhead. We have proposed a systematic approach to threat identification starting from the analysis of the activities in the use cases of the system and postulating possible threats [Bra08]. This method identifies high-level threats such as "the customer can be an impostor", but once the system is designed we need to see how the chosen components could be used by the attacker to reach her objectives. For this purpose we proposed the use of *misuse patterns* (which we called initially attack patterns) [Fer07]. A misuse pattern describes, from the point of view of the attacker, how a type of attack is performed (what units it uses and how), analyzes the ways of stopping the attack by enumerating possible security patterns that can be applied for this purpose, and describes how to trace the attack once it has happened by appropriate collection and observation of forensics data. It also describes precisely the context where the attack may occur. We built a catalog of misuse patterns for VoIP [Pel09] and we characterized precisely some aspects of misuse patterns [Fer09]. We describe this type of patterns using a template based on the one used in [Bus96], which is commonly used for architectural patterns as well as security patterns. This catalog is not only useful to test a new system but also to evaluate an existing system.

To make misuse patterns of practical value we need a catalog of typical attacks. As we said above, until now we have only misuse patterns for VoIP environments, this is our first misuse pattern of a more general scope.

Worm

Intent

Propagate to as many places as possible (or to specific systems), usually indicating its presence, and maybe performing some damage.

Context

Sites connected through the Internet or another type of network. The Internet provides a variety of services such as email, file transfer, and web services (Figure 1). Any of these services can be used for propagation. Both fixed and wireless networks can be used by the worm. Portable storage devices such as memory sticks can also propagate worms.

Problem

A worm tries to take advantage of any input to invade a system. Users might open attachments carrying worms and some ports of a system may be unprotected or have vulnerabilities; all of these give the worm a chance to invade. Mail systems and file transfer systems for example, include lists of addresses which can be used by the worm to find places where to propagate. Many systems do not control access to their system directories and do not restrict Internet traffic, which facilitates a worm invasion.

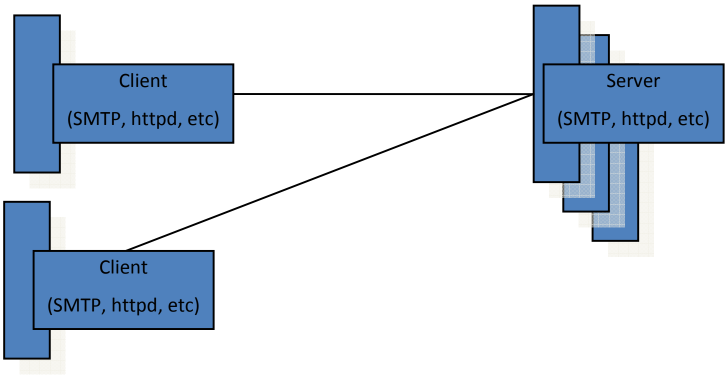

Figure 1. Context for worm propagation

The solution is affected by the following **forces :**

- *Objectives*. Its objectives may be political, monetary, or vandalism. A political worm typically tries to produce damage to an antagonist; a monetary worm tries to reach many places to collect information or drop spyware; a vandal worm tries to destroy or damage information.

- *Reach.* Try to reach as many places as possible or to specific sites. For most worms, reaching many places is a basic objective.

- *Presence manifestation.* Try to show its presence in the system so victims know about it. Exceptions to this are cases where the objective is to drop spyware.

- *Credit.* To embed an identification or mark so that the creator can take credit for it.

- *Misuse.* Perform some destruction and/or other misuses (confidentiality, integrity, or availability). The misuse may be delayed (time bomb).

- *Obfuscation.* Try to hide its structure to make harder its detection and removal.

- *Collateral damage*. In addition to specific misuses, the worm may require costly operations for its removal, stopping or disrupting business activities. Its propagation may affect the normal traffic in the network.

- *Latency.* Its propagation must be as fast as possible to avoid detection and countermeasures.

- *Activation.* This can be done by *e*nticing offers which may tempt users to open email attachments or download procedures (social engineering). Other possibilities are invading through unprotected ports or taking advantage of vulnerabilities.

Solution

Attach a core portion of the worm to email messages or to files. When the user opens the message attachments or executes the file the core of the worm starts executing. Alternatively, invade through an unprotected or flawed port. Download remaining portions from complementary network sites. Use some procedure to hide the structure of the worm. Perform its mission and propagate. Figure 2 shows the propagation of a typical worm; speed comes from a tree-like propagation.

Structure

Figure 3 shows a class diagram of the units involved. Class **Node** represents any node in the network, defined by its address (URL in the Internet). Any node can be the *origin* of a worm and any node can be its *target* (and be invaded). Some nodes are *complementary sites* from which commands or other parts of the worm may be retrieved. Class **Worm** represents the worm itself, including procedures for initial setup, to bring complementary parts, to hide the worm, to perform its mission, and to propagate.

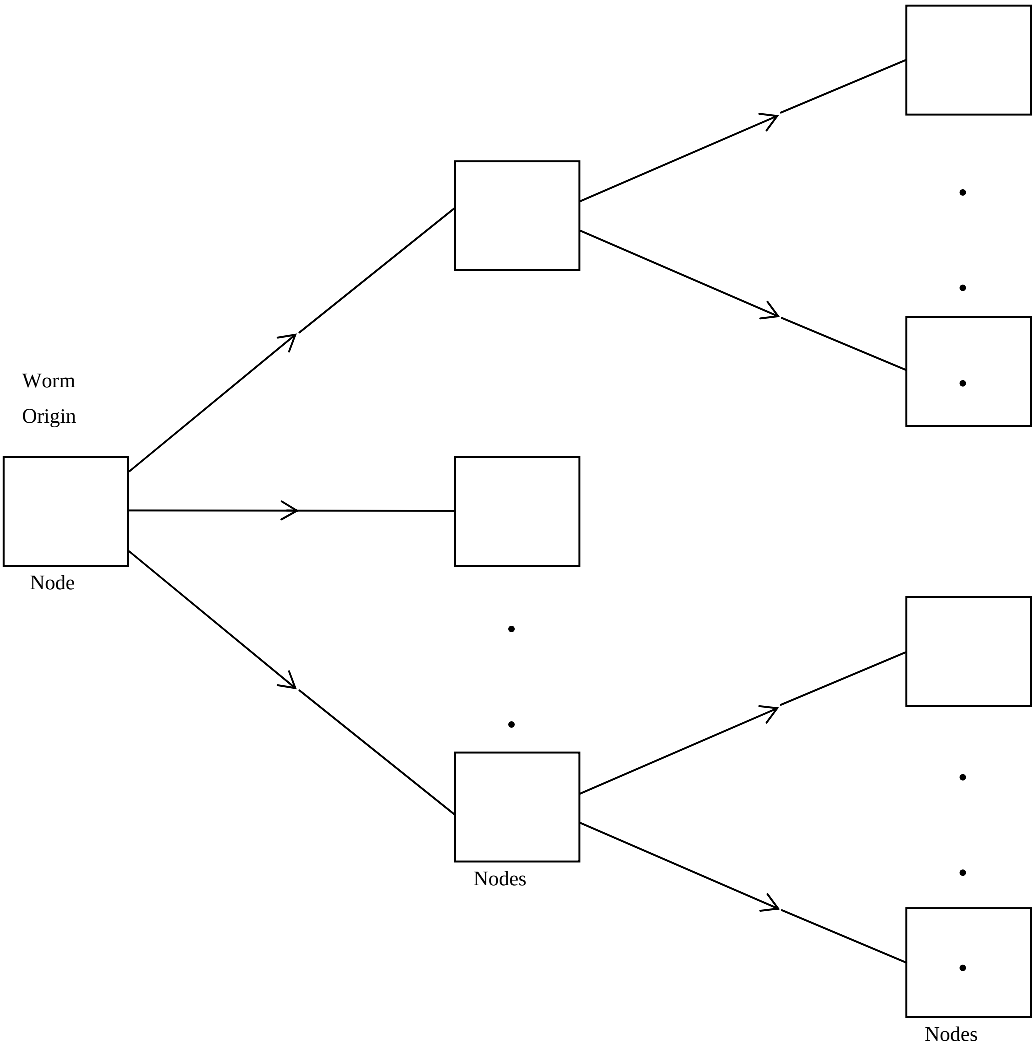

Figure 2   Worm propagation

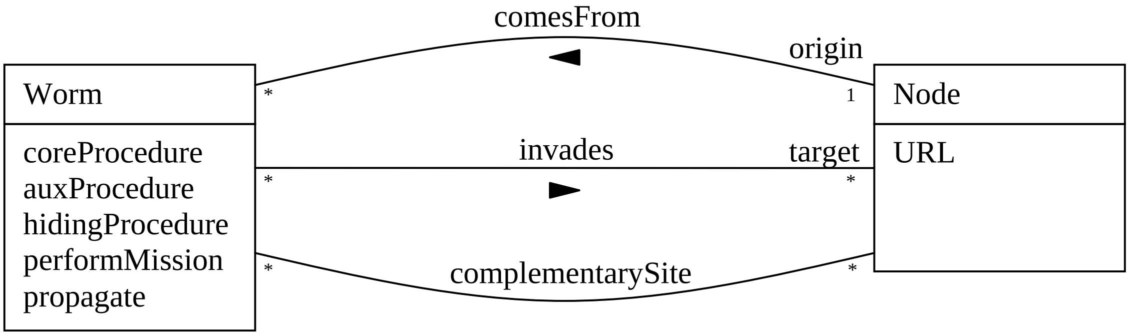

Figure 3. Class diagram for the Worm pattern.

Dynamics

Use cases for a worm may include Create a Worm, Remove a Worm, and Activate a Worm. Create and Remove are specific to the type of worm (see Variants). We describe here Activate a Worm because it is the most important for defenders. Its scenario (Figure 4) includes:

- *Triggering:* After the attacker sends a message, a target (user) may activate an executable procedure with a core part of the worm.
- *Assembly:* Download remaining parts via the Internet (optional)
- *Obfuscation*: Use some procedure to hide the parts of the worm, e.g. encryption or dispersion.
- *Address Search*: Find destination addresses as new targets for propagation. Addresses may also be generated randomly.
- *Manifestation*: Display some messages (optional)
- *Propagation:* Send the core part via the connection to another node in the address list. This operation is repeated for all the found or generated addresses.

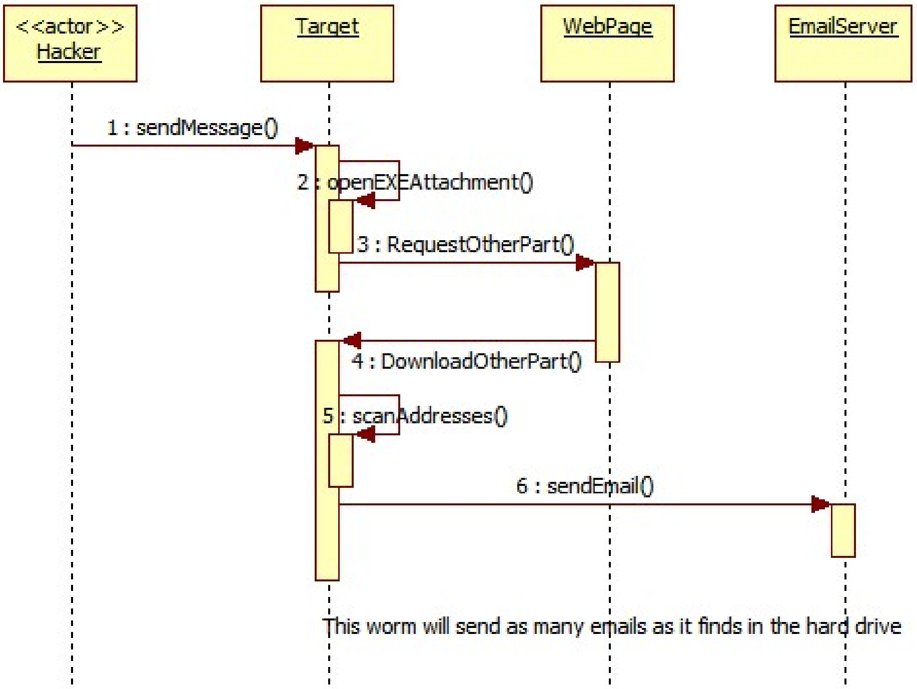

Figure 4. Sequence diagram for activating a worm

Variants

A *passive worm* requires a user to activate an executable program and it usually propagates through email. Melissa, ILOVEYOU, Anna Kournikova, and Bagle are examples of this type.

An *active worm* takes advantage of some system flaw to provoke a buffer overflow or another attack to get in through some port. It may scan looking for unprotected ports. Code Red is an active worm. Storm can be active or passive [Smi08].

A *virus* attaches itself to some program (infects an executable file) and when the user executes this program it gets activated. Jerusalem, Christmas, and Chernobyl are examples of viruses.

Some worms have several versions with different purposes; for example, Storm has variants that perform different types of misuses, including targeted spam and DDoS attacks [Smi08].

Some worms are *multimode* (multivector) worms, which can use a variety of ways to invade their targets; for example the Storm virus infects computers using multiple payloads [Smi08].

Known uses

Typical examples of worms include:

- *ILOVEYOU* [ILO, wor09]. This was an email attachment worm that appeared in 2000. It relied in social engineering to entice users to open the attachment. It also used specific weaknesses of Microsoft Windows. It propagated using the addresses in the address book of the mail system.

- *Bagle*. It was a mass-mailing worm written in assembly language [bag] and affecting all versions of Windows. After activation, it copies itself to the Windows system directory and downloads a SMTP engine to mail its core to other nodes as an attachment (see the Implementation section for its typical behavior).

- *Code Red* [Ber01]. It appeared in July of 2001. It propagated through port 80, indicated its presence by defacing web pages, propagated using a random IP address generator, and later would activate a denial of service attack from infected sites.

- *Nimda* [nim]. Nimda is a multivector worm that can use several ways to propagate: email, visiting an infected site, seeking out vulnerable servers to upload files, or through the network.

- *Slapper* [Arc03]. Can launch denial of service attacks. Propagates finding addresses in files. The nodes invaded by the worm communicate using a P2P protocol to collaborate in their misuses.

- *Conficker* [con09, wor09]. This is a multivector worm with an autoupdate facility (signed updates) and encrypted communications. It downloads parts of the worm from some Internet sites.

These worms are really worm types from where many variants can be derived. It is possible to define separate patterns for each type of the generic Worm pattern. For example, the Slapper worm and the Apache Scalper operate in a similar way [wor09], the Conficker is really a series of worms [wor09].

Implementation

We show a typical implementation of the Bagle worm. It follows very closely the sequence diagram of Figure 4. A scenario in a Microsoft environment would include:

- A user invokes an executable code by clicking a MS Word file, then automatically VBA macro code is interpreted.
- The worm downloads the remaining parts from a web server via the Internet.
- The worm finds target addresses in the Outlook address book using VBA and a SMTP server name from outlook settings.
- The worm displays some messages using a VBA function.
- The worm opens a SMTP connection to mail its core to the next target. This operation is repeated for all the found addresses.

Active worms take advantage of vulnerabilities such as buffer overflows and can get in through port 80 or unprotected ports. In the case of worms such as Code Red the core of the worm was sent to the input buffer of port 80 in Microsoft's IIS server [Ber01]. A virus or worm may send a web address link as an instant message to all the contacts of the invaded site and if the recipients answer, they bring the virus to their sites.

Consequences

This misuse has the following *advantages* for the attacker:

- *Objectives*. Its economic objectives can be reached if the worm has a long reach and clever social engineering. Its political objectives can be reached if the worm reaches the intended audience and manifests its presence and reasons. Its vandalism objectives can be obtained if the worm does considerable damage.

- *Reach.* If the system has easily accessible address lists the worm can find many new targets. Random address generation is not so effective.

- *Manifestation of its presence.* A good procedure for display can make its presence well noticed. This may intimidate its victims, which brings satisfaction to the attacker.

- *Credit.* The mark should be distinctive but not identify the attacker. The creator can get negative recognition for his effort.

- *Misuse.* A worm can perform destruction and/or other misuses (confidentiality, integrity, denial of service, drop spyware or spam).

- *Obfuscation.* Encryption and dispersion can make harder its detection and removal. Some worms mutate, i.e. they change their structure when they propagate.

- *Side effects.* A fast-propagating worm can produce a lot of traffic and if it is hard to detect its cost increases.

- *Latency.* A fast-propagating worm can do much damage before being stopped.

- *Activation.* Good ways to activate the worm are necessary since all its objectives depend on this step.

A worm also can have some *liabilities* for the attacker:

- A worm can be used to detect infected nodes or to destroy viruses or other worms.

Countermeasures

The following policies and their corresponding mechanisms (realized as patterns), can stop or mitigate the worm:

- *Policy about attachments*: Users should be trained to recognize trustable attachments and they should be forbidden to open unknown or suspicious attachments.

- *Need-to-know* policy to define access by system processes to resources. For example, address lists should use authorization to control access to their contents.

- *Control of network communications*: Connections should be established with only trusted addresses (control through the firewalls). This policy may avoid downloads from complementary sites.

- *Intrusion detection:* An IDS can detect some attacks in real time and alert the firewall to stop it.

- *Use of antivirus software*: Can help detect and clean worms after the fact

- *Backups.* Checkpointing files and keeping backup images of them is a fundamental precaution against data destruction or unauthorized modification.

- *Specialized hardware.* Process communication controls in the operating system can be enforced through specialized hardware [Shi00]. It is possible to define partitions in the operating system that can be enforced by hardware and will prevent a worm from performing its actions.

Forensics

The pieces of the worm may be scattered in different units within a site. The specific places to look for worm components depend on the specific variant or type of worm. The places where worms normally penetrate include mail attachments, files, unprotected ports, and these must be inspected. One should also look for the specific parts of the work, e.g. core procedure, obfuscation procedure, etc.

Web logs can help in finding parts that might have been downloaded. GUIs may have log records of the use of procedures to display the worm announcements. Units that contain addresses may contain indications of search.

Related patterns

- *Authorization and Reference Monitor*. These patterns together can prevent access to address lists, thus stopping the worm propagation [Sch06].
- *Firewall*. Can filter attempts to download further pieces of the worm [Sch06].
- *Intrusion Detection*. Can detect a worm invasion in real time and collaborate with the firewall to block its traffic [Fer05].

Acknowledgements

We thank our shepherd, Tsukasa Takemura, for his useful comments that significantly improved the quality of the paper. We also thank Eiiti Hanyuda for supervising our paper shepherding.

References

[Arc03] I. Arce and E. Levy, An analysis of the Slapper worm", *IEEE Security and Privacy*, Jan./Feb. 2003. 82-87.

[bag] "Bagle (computer worm), *http://en.wikipedia.org/wiki/Bagle_(computer_worm)*

[Ber01] H. Berghel, "The Code Red worm", *Comm. of the ACM*, vol. 44, No 12, December 2001, 15-19.

[Bra08] F. Braz, E.B.Fernandez, and M. VanHilst, "Eliciting security requirements through misuse activities" *Procs. of the 2nd Int. Workshop on Secure Systems Methodologies using Patterns (SPattern'07*). In conjunction with the 4th International Conference onTrust, Privacy & Security in Digital Business (TrustBus'07), Turin, Italy, September 1-5, 2008. 328-333.

[con] "Conficker", http://en.wikipedia.org/wiki/Conficker

[Fer05] E.B.Fernandez and A. Kumar, "A security pattern for rule-based intrusion detection", *Proceedings of the Nordic Conference on Pattern Languages of Programs, Viking PLoP 2005*, Otaniemi, Finland, 23-25 September 2005.

[Fer07] E.B. Fernandez, J.C. Pelaez, and M.M. Larrondo-Petrie, "Attack patterns: A new forensic and design tool", *Procs. of the Third Annual IFIP WG 11.9 Int. Conf. on Digital Forensics*, Orlando, FL, Jan. 29-31, 2007. Chapter 24 in *Advances in Digital Forensics III*, P. Craiger and S. Shenoi (Eds.), Springer/IFIP, 2007, 345-357.

[Fer09] E.B. Fernandez, N. Yoshioka and H. Washizaki, "Modeling misuse patterns", *Procs. of the 4th Int. Workshop on Dependability Aspects of Data Warehousing and Mining Applications (DAWAM 2009),* in conjunction with the *4th Int.Conf. on Availability,* Reliability, and Security (ARES 2009). March 16-19, 2009, Fukuoka, Japan.

[ILO] "ILOVEYOU", http://en.wikipedia.org/wiki/ILOVEYOU

[Nim] "F-Secure Virus-descriptions:Nimda", http://www.f-secure.com/v-descs/nimda.shmtl

[Pel09] J. Pelaez, E.B.Fernandez, and M.M. Larrondo-Petrie, "Misuse patterns in VoIP", *Security and Communication Networks Journal*. Wiley, published online**:** 15 Apr 2009 http://www3.interscience.wiley.com/journal/117905275/issue

[Sch06] M. Schumacher, E. B.Fernandez, D. Hybertson, F. Buschmann, and P. Sommerlad, *Security Patterns: Integrating security and systems engineering,* Wiley 2006.

[Shi00] T. Shinagawa, K. Kono, T. Masuda, " *Exploiting Segmentation Mechanism for Protecting Against Malicious Mobile Code*", Tech. Report 00-02, Dept. of Information Science, University of Tokyo, May 2000.

[Smi08] B. Smith, "A Storm (worm) is brewing", *Computer*, IEEE February 2008, 20-22.

[wor09] "Worm evolution", May 2009, http://www.digitalthreat.net/?p=17