An Overlay Network for Monitoring Activities of Computer Viruses

Li Ying[1,2], Cao Yiqun[2], Qiu Ben[2], Jiao Jian[2], Shan Xiuming[2], Ren Yong[2]
(1. School of Business, SUN YAT-SEN University, Guangzhou 510275, China;
2. Department of Electronic Engineering, Tsinghua University, Beijing 100084, China)

Abstract: To accurately track computer viruses, an overlay network that monitors the activities of viruses is constructed. Identifying and locating nodes infected by virus on network is achieved by a naming system in which a node in the network is mapped to a unique serial number of the hard-drive. By carefully monitoring and recording sensitive communication between local system and remote nodes on the network, and suspicious operations on files that originate from remote nodes and entered via some form of file transfer, activities of viruses in both local and network level are recorded and ready for future analysis. These data can also be used in analysis of the mechanism of a computer virus as well as its spreading mode and pattern.

Keywords: overlay network; virus; observation; DNS

1 Introduction

The development of computer network and internet provides an ideal environment for the spread of computer viruses. Slammer (also named Sapphire), which occurred in January of 2003, prevailed so quickly that the number of machines infected doubled every 8.5 seconds, and reached 90% servers globally with corresponding vulnerability within 10 minutes[1]. Using Internet, computer viruses now spread at an astonishingly increasing speed, making the work of anti-virus much more difficult. On the other hand, the fast spread of computer viruses over the network consumes computing and communication resources, largely degrading the efficiency of the network. Recently the spread of Blaster, Welchia, and Sobig Worm have all caused congestion in a large scale. So tracking the activities of virus from a network point of view, makes much sense to the detection and recognition of virus, to tracing virus source, and to determining the paths along which a virus spreads. It can also provide valuable information for killing virus along the propagation path completely and exactly in the future.

Overlay network, which is an abstract network built on current net, is regarded as the trend of network technology. It is usually designed to implement a certain function intended to improve the performance of network. QoS overlay, routing overlay, security overlay are typical examples of overlay networks. There are several research projects that apply overlay network to address DDoS problem. For example, CenterTrack[2] can inspect the packets which include mendacious source addresses, and SOS is able to protect network from DDoS[3,4] attack. However, there has been no overlay network aiming at stopping the propagation of computer virus over the computer network or securing computer systems in a network from a network perspective yet.

We design an overlay network to monitor the activities of viruses based on the dynamic properties of viruses. This overlay network allows locating infected network nodes accurately, and monitoring activities from both a local perspective and a network point of view. The recorded activities cover the sensitive communications between remote nodes on the network and the local system, and suspicious operations on the files that originate from remote nodes and have entered the system via some form of file transfer. All these data can be used to analyze mechanism and spreading mode and patterns of viruses, and to provide important hint to further securing computer systems from both local and network perspetives.

The rest of the paper is organized as follows: Section 1 discusses the principles of virus spreading model. In Section 2, the outline of the overlay network is introduced. The details in technological realization are discussed in Section 3 with conclusion presented in Section 4.

Received 29 *December* 2006

2 The Principles of Virus Spreading and Spreading Model

2.1 The Principles of Virus Spreading

Network viruses spread mostly in the following ways: a. through e-mails. Virus either exists in attachments, deceiving the user to execute it by social activities, or directly hides in the content, accomplishing its goal by exploiting bugs in the email client programs. For example, the infamous ILOVEYOU worm spreads as e-mails, and by deceiving the user to click on a malicious attachment, it will send itself to all contacts in the user's address book. b. Through web browsing. Malicious code is embedded into the web pages and is executed in some vulnerable web browsers. For example, the Bofra worm exploits the IFRAME vulnerability of Internet Explorer to enter the user's system and get activated immediately. c. Through file sharing. This is getting more and more common, with ubiquitous file sharing introduced by NetBIOS over TCP/IP, and also the users' insufficient knowledge on computer security. For example, Nimda worm uses NetBIOS over TCP/IP as one way to spread itself by infecting accessible shared files. d. Through other network applications. Almost all real-time communication software can deal with file attachments or file sharing, which may both be harnessed by viruses. Also, in network servers, the server software is also a common target. For example, Henpeck worm spreads through MSN Messenger network.

Several conclusions are summarized as follows:

1) Most viruses spread in the form of file transfer. Viruses enter and exit local systems mainly in file form. When a virus spreads via E-mail attachments, the attachments will be hold as temporary files before their malicious function is activated. Malicious web pages are also first saved as temporary files before they are interpreted. And the virus that spreads through file sharing also exists as a part of the infected file. Even though some viruses may enter systems in some other means, sometimes not directly in file form, but to complete the infection, they commonly create new files or infect existing files. Therefore, the spread of most viruses depends on file transfer, including responding to resource request through HTTP protocols, attachments in E-mails, and files transferred in file sharing protocols.

2) Port program. The programs that handle file transfer are called port programs. By blocking these programs, computers can not transfer files across the network—most network viruses will be eliminated thoroughly, and most normal network applications cannot function, either.

3) Trustworthy and untrustworthy file. Trustworthy files are secure local files, while untrustworthy files mean temporary files originating from other nodes in the network and enter the local system through port programs and whose security has not been approved yet. System should not execute the code in untrustworthy files, because malicious code may exist in them. Untrustworthy files are often introduced to local system because of current implementation of distributed system - a proxy to remote resources is often implemented using local cache. Therefore, untrustworthy files can always be seen as temporary files. It is noted that whether a file is trustworthy or not is often not decided by the user, but by local system according to some rules. Take web browsing as an example, when a web page is loaded, a local caching file will be created, and the web browser will estimate its security level in order to supply appropriate execution permission. Untrustworthy files can be relabeled as trustworthy files, according to either rules in the local system, or explicit commands from users.

Take web browsing as an example again. The Internet Explorer, which is considered as the most widely used web browser, allows automatic download and installation of browser plug-ins. If the Internet Explorer is set to a specific security level, it will prompt the user before installing a plug-in required by the currently loaded web page. Essentially, this is a file transfer process, followed by an attempt to relabel an untrustworthy file as trustworthy. If the security level is set to a low level, the Internet Explorer will probably complete the file transfer process and relabeling automatically without asking for any confirmation from the user. With a security level high enough, the required but missing plug-in will be ignored completely by the browser. Consequently, potentially untrustworthy files, as well as potentially trustworthy files, will have no chance to enter the local system.

The virus infection is therefore a process in which some untrustworthy files get transferred and improperly relabeled as trustworthy either by taking advantage of vulnerabilities of port programs or by deceiving users, and then get activated.

4) Active and inactive virus. Even if a computer virus has entered a local system and been labeled as trustworthy files, it still exhibits two different states, active and inactive. Only an active virus is effective. An inactive virus can be activated through several different ways. First, it may occur because of some system vulnerabilities. For instance, some viruses make use of MIME bug of the Internet Explorer to activate a

malicious executable. Second, activation can also be accomplished by changing system settings. For instance, most Trojan horse programs change settings in system registry to activate malicious executable automatically on system startup. Finally, deceiving users performs well all the time. For example, the infamous Nimda used a malicious file named readme. eml and managed to trick millions of users to activate the malicious code hidden in it.

2.2 The Model of Virus Spread

A model of virus spread can be abstracted from the above analysis and is presented as follows. A virus in the form of files enters the local system through port programs, gets labeled as trustworthy, and is finally activated to start a new cycle of propagation. Therefore, improvement of port programs and limitation of activation will be useful to prevent virus spreading.

1) Improvement of port programs. Vulnerabilities of port program play a crucial role in security problem. In recent years more and more vulnerabilities have been discovered, and common port programs have received more frequent updates. Users are advised to use the latest versions of these programs to avoid historical problems. Applying update in a timely fashion is also important.

Improperly classifying files as trustworthy by inexperienced users are dangerous, too. Currently, most PC users lack knowledge on computer security, so it is not realistic to demand them to always correctly distinguish untrustworthy files from trustworthy ones. In this situation, the default settings should be secured enough for the most common users. At the same time, adequate advice and warning should be provided at a critical circumstance when manual work is necessary. Outlook 2002 does well in alerting users, and an attempt to execute macro code in attachments will trigger a warning.

2) Disabling path to activation. Virus spreading can be stopped without the virus getting activated. Viruses are computer programs, as non-malicious normal programs. Therefore, the system cannot function normally if all approaches of activation are disabled, because that will also disable normal programs, which make a computer is useful in the first place. We believe that three kinds of work which makes activation harder are desirable: first, the vulnerabilities of system must be patched in a timely fashion; second, warning messages and/or confirmation should be triggered when a suspicious and/or dangerous operation is detected; finally, some critical system settings, such as critical entries in system registry, which often used by virus as a path to activation should be monitored.

3 An Overlay Network for Monitoring of Activities of Virus

An overlay network for anti-virus purpose is proposed in the context of today's network-based viruses. With the development of computer and network technology, new kinds of viruses emerge continuously. The puzzle of the border between viruses and normal programs is fuzzy and there seems to be no solution to it. Nowadays no anti-virus technology or system can promise to defeat all threats to computer and network security. We introduce an overlay network that aims at a powerful security enhancement to current technological offerings.

Based on the spreading model above, main functions of this system include: a. monitoring activities of port programs, mainly the transfer of files, both into and out of the local system; b. recording suspicious activities of executable files originating from network; c. analyzing the monitoring data to determine the infection path and malicious operations when a file is categorized as a virus or virus carrier; d. inter-node communication to support the overlay network. In summary, the system can identify every node uniquely and locate every node exactly, and observe and record suspicious file transfer on network and suspicious local activities of files originated from outside. All functions are illustrated in Fig. 1.

As illustrated in Fig. 1, a naming subsystem assigns a name to each node of the network. The activities monitoring module observe sensitive activities inside the local system and at the boundary of local system. Monitoring module will record activities to a central database. A data analysis module will read the database and extract useful information, which will be used to create security rules to further enhance the security of the system and the network as a whole.

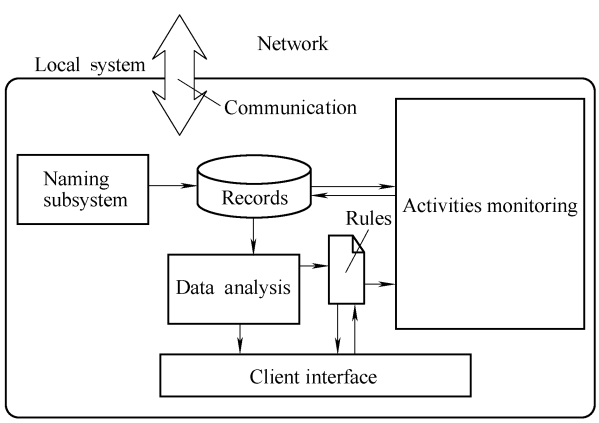

Fig. 1 Functions of the system

4 System Architecture

The modules and their functions are described as follows.

1) Naming subsystem and communication module. Naming subsystem forms the network name space for nodes in this overlay network. It assigns a unique ID for each node, which is used in all types of records. Communication module traces the file transfers, and record corresponding information for later analysis. The two parts are put together because the naming subsystem is mainly used for recording file transfers on net.

2) System monitor and record module. This module tracks local activities, i. e. file exchange between the local system and the network, as well as suspicious operations of files originating from network. As discussed in Section 1, most viruses pass in and out of the local system in the form of files, so recording the file exchange is necessary. Information recorded includes the original source of the file, the path under which the file is saved in the local system, the ID of the destination node if it is transferred out, etc. The information on activities of suspicious files will later be used to recover the activities of potential virus. All programs that originate from the network, before classified and labeled as trustworthy by system according to some rules or manually, are regarded as suspicious files. Once the system detects that a suspicious program starts, it will monitor its behaviors, record all files written by it, and add these files to the list of suspicious files. With this module, all activities of a virus infecting a local system, from its entrance, replication, infection, and destruction to passing out, are reflected in the records. With the help of naming subsystem, various types of information can be inferred, such as whether it is a virus, what kind of virus, and where it is from and where it goes.

3) Data analysis module. when a program is identified as a virus, this module will compute its source, destination and all behaviors by analyzing the records. The result will also be fed back to monitoring strategy and other security rules.

4.1 Naming Subsystem and Communication Module

To uniquely identify and exactly locate every node on the network, a new naming system with DNS structure is constructed.

A desirable identification to the network nodes should have the following three basic properties, a. uniqueness, b. consistency, and c. addressability. A unique ID ensures a one-to-one mapping at any time. A consistent identification will not change over time, allowing a node to be identified later based on name listed in previous records. An addressable identification means that the identification can be used to look up a network address, which can be used to establish network data connection. For example, IP address as an identification is often unique and addressable, but not consistent since the mapping from an IP address to a physical machine can change. So IP address cannot be used here, and a new naming subsystem is needed.

One solution would be to build a new naming system from scratch, but that involves much work, including design of new protocols, new client and server programs, and concerns on security and future update. Instead, we base our naming system on Domain Name System (DNS). Taking advantage of the hierarchical structure of DNS, our naming subsystem is a subtree in the global DNS name tree.

Our naming subsystem assigns an identification name (ID) to each hard-drive. As mentioned above, viruses are hosted in files in a local file system, which is then physically carried in a local hard-drive. Therefore, we select hard-drives to be nodes of the overlay network. An ID is constructed for each hard-drive based on its unique serial number. Such an ID is unique and consistent. The DNS is used to map such an ID to the IP address of machine currently hosting the hard-drive, making this naming scheme also addressable. The properties of uniqueness, consistency and addressability allow it to be used in tracking virus propagation, identifying and locating infected nodes, and tracing the source of the virus on network. The robustness and security of this subsystem is guaranteed by those of the global DNS. Hierarchical structure of DNS implies that we can also segment our network into subnetworks in a hierarchical fashion.

For implementation, a DNS domain name is required, and ID to IP address mappings will be hosted under this domain name. We registered a domain name "nsas. ipchina. org" for our implementation. This domain will host all the mappings from ID to IP address for our naming subsystem, in a flat fashion if there is no subnetwork or in a hierarchical fashion if the network is divided into subnetworks for the sake of scalability. In the current implementation, there is only one central node, named "ns1. nsas. ipchina. org", where the server of the naming subsystem is installed. When network grows large, the domain can be divided into subdomains, each of which serves a subnetwork of the network. This will effectively reduce system load thanks to scalable design of DNS. The task of the cen-

tral node is to maintain a database for the name lookup based on the mapping between IDs and IP addresses. Since it is essentially the same as DNS lookup, we can use conventional DNS server, such as BIND. If a host has more than one hard-drive, it will have corresponding IDs mapped to the same IP address.

Similar to DNS, our naming system needs a client program in addition to a server program. The main function of a client program is to read the serial numbers of hard-drives, send them to the servers, and keep the server updated on changes of associated IP. The servers receive the reports from clients, and update database accordingly. Keeping the mapping up-to-date is essential to the effectiveness of the whole system. When a file transfer takes place, the unique IDs of the both ends of the transfer will be recorded. An up-to-date mapping from IDs to IP addresses allows addressing these two nodes at any time later even if their IP addresses have changed, and more operations such as sending anti-virus information and software update to be possible.

4.2 Monitoring of Local Activities

Monitoring local activities aim at accurate and complete tracking of suspicious files. Recording transfer of suspicious files is not sufficient in this sense, because a non-suspicious file can potentially be transformed into a suspicious one. When a suspicious file carries virus, and the malicious function of the virus gets activated locally, it may infect other files. As a result, the infected files, although not originating from outside, become new suspicious file. So in addition to keeping track of suspicious files which originate from network, it is necessary to monitor the activities of the executable suspicious file, and files modified or created by a malicious executable file must be added to the list of suspicious files.

The monitoring mainly consists of two tasks. One is to monitor the creation of new processes and to check whether it is created from suspicious files. When a new process is created from an untrustworthy executable file, the process is untrustworthy. The other monitoring task is to track untrustworthy process, and particularly when write operation occurs, record this operation and update suspicious file list by some rules. A possible simple rule is based on file name extension. For example, if the created file has name ending in .exe or .hml, it will be added to the suspicious file list.

This module runs on the Windows platform, particularly Windows NT platform, including all major Windows platforms widely used at the present time. The platform offers favorable application programming interfaces (API), and lots of tools and documents, which simplify the development of this module.

Our implementation of the monitoring module has borrowed ideas and components from many existing proposals and products. API Spying Techniques for Windows [5] is of particular interest to us. Our implementation also uses the Detour Programming Packages [6]. Based on the core function of tracking and intercepting interesting API calls, we are able to add many useful functions, such as fine-grained control of file access, process creation, etc. These together form a fully functional and easily extendable monitoring module and a platform to enforce other fine-grained controls.

In addition to the function of monitoring and recording local activities for later analysis, there is some extra benefit introduced by this module. Being able to monitor activities in real time makes it possible to implement Heuristic Techniques [7], and to prevent some malicious behaviors, such as access to some sensitive files and registry entries.

Extra effort has been put to defense against script viruses. Script viruses differ from traditional ones based on binary executable in many ways. Generally, script is simpler but powerful, and is getting more and more popular. Script programs are interpreted when running, which makes monitoring activities at the process level less effective. Besides, the fact that various interpreters is used introduces more complexity. We start with Microsoft Active Script Technology [8], which is a relatively common script interpreter used in many Microsoft products. We analyzed the principles of script interpretation in this environment, and proposed a scheme to monitor activities of scripts based on COM [9] and Automation Technology [10].

4.3 Monitoring of Network Activities

Monitoring of network activities is to record file exchanges, including file transfer via email, web navigation and file sharing via LAN. A separate module is created to handle this task.

Files can be passed into and out of local systems in many ways. What makes it worse is that people are free to create new implementations of existing protocols, or even new protocols, to enable new ways of file transfer. So it is difficult, if not impossible, to track all file exchanges between local systems and network accurately and comprehensively. A relatively more comprehensive monitor can be built at the network layer, by examining all IP packages to build a complete picture of communication between the local system and the network, as some firewalls do. However, since we aim at monitoring file transfers, we have to collect IP packages to recover a high level understanding of file transfers on the application layer, which is a complex if

not impossible. Fortunately, a number of features are shared among most virus files. First, their spreading is autonomous, with little interference by the users. Second, they highly depend on specific Operating Systems and platforms commonly found to achieve large-scale infection, and therefore mainly use several built-in network programs included with Windows to transfer files. Based on the analysis above, it is effective to just monitor these built-in programs, such as Internet Explorer as web browser, Outlook as email client, and monitor their network activities.

Implementation of this monitoring module requires modification to the aforementioned programs to include activities tracking functions. Thanks to the extensible design of these programs, we implemented this module based on technologies such as COM [9], BHO [11], and MAPIHOOK, by creating module to attach to original programs, and without direct change to those programs.

The monitoring module records activities to a database using ADO Technology, which simplifies the task of data management. By using Microsoft Access format for the database, a user or a data analysis program can access the database using all normal database operations.

4.4 Data Analysis

Data collected by the monitoring module consists of sequence of unorganized local events and node-to-node interaction in the network, and there is no direct hint on the path along which a virus spreads. A data analysis module is created to mine these data to extract useful information.

The analysis module works at two different levels. The first is the local level, and it concerns which files may have been infected and the mechanism of virus functions. The other is the network level, which aims at telling how virus achieves point-to-point propagation, which nodes have been infected, the topology of the affected subnetwork, and how its effectiveness correlates to the topology and other properties of the overall network. Analysis at a local level can help local systems to defend against similar viruses by abstracting a point of attack and improving security accordingly. Analysis at a network level can help determine the paths, speed and the scale of virus spreading. This part is still under further design and development, and here we will give two examples to illustrate the utilities.

Data analysis at the local level can effectively improve system security against attack by viruses similar to known ones. When a local system is infected with a virus, before it is categorized as virus manually either by the user or by anti-virus service provider, it exists in local system as a normal program. When it is categorized as virus, traditional anti-virus solution only provides defense against future infection of virus of exactly the same kind and is not very effective in dealing with variants. With our system, data collected by monitoring module can help predict sensitive system files and/or registries harnessed by this virus, and by enforcing protection with the monitoring module for local activities to secure these sensitive files and/or settings, it is much harder for a variant to infect this system.

Data analysis at the network level can help secure computer systems of an organization as a whole. By analyzing the spreading path of a virus, we can distribute anti-virus packages and organize anti-virus effort in a more efficient and effective way. For example, when resources are limited, we may want to rescue the nodes which are at the highest risk of getting infected. We can also potentially find super infectors, i. e. nodes which spread the virus to many other nodes. Such super infectors are weakness in the security chain of the whole network, and may need more attention from security personnel. By looking at security at the network level, an organization can secure its computer systems and network much more effectively.

5 Conclusion and Future Work

The overlay network for monitoring of activities of virus is constructed by taking the serial number of a hard-drive as the ID of a network node, and building a Naming System using existing DNS infrastructure [12]. The network can accurately identify and locate infected computers. Sensitive activities data, including suspicious communication between local system and network and suspicious operations of untrustworthy files originating from, are monitored and recorded. Analysis on these data can help understand the mechanism of virus functions and defend against similar viruses in both the local and the network level, and can also provide important hint to further securing computer systems and networks.

Future work will mainly focus on analysis of monitoring. The ultimate goal of this system is to track and kill viruses along their spreading paths, and to identify similar viruses at the network level. Therefore, analysis module that is capable to recover the spreading paths and patterns from monitoring data will be created. Information provided by this module will be used to help organize anti-virus efforts and identify similar viruses by their activity pattern on a network level.

References

[1] Moore D, Paxson V, Savage S, et al. The Spread of the Sapphire/Slammer Worm[EB/OL]. http://www.caida.org/outreach/papers/2003/sapphire/, 2006

[2] Stone R. CenterTrack: An IP Overlay Network for Tracking Denial-of-Service Floods [EB/OL]. http://www.nanog.org/mtg-9910/robert.html, 1999

[3] Keromytis A D, Misra V, and Rubensten D. SOS: secure overlay services[A]. Proceedings of ACM SIGCOMM [C]. Pittsburgh. 2002

[4] Wang Ju, Chien A. Using Overlay Networks to Resist Denial-of-Service Attacks [EB/OL]. http://citeseer.ist.psu.edu/wang03using.html, 2003

[5] Yariv K. API Spying Techniques for Windows 9x, NT and 2000 [EB/OL]. http://www.internals.com/articles/apispy/apispy.htm,2000

[6] Galen H, Doug B. Binary interception of Win32 functions[A]. Proceedings of the 3rd USENIX Windows NT Symposium[C]. Seattle, Washington, 1999

[7] Markus S. Heuristic Techniques in AV Solutions: An Overview [EB/OL]. http://www.securityfocus.com/infocus/1542, 2002

[8] Microsoft Windows Script Technologies [EB/OL]. http://msdn.microsoft.com/library/default.asp? url=/library/en-us/script56/html/vtoriMicrosoftWindowsScriptTechnologies.asp,2006

[9] The Component Object Model: A Technical Overview[EB/OL]. http://msdn.microsoft.com/library/default.asp? url=/library/en-us/dncomg/html/msdn_comppr.asp,2006

[10] Overview of Automation [EB/OL]. http://msdn.microsoft.com/library/default.asp? url=/library/en-us/automat/htm/chap1_3rlq.asp,2006

[11] Browser Helper Objects: The Browser the Way You Want It[EB/OL]. http://msdn.microsoft.com/library/default.asp? url=/library/en-us/script56/html/vtoriMicrosoftWindowsScriptTechnologies.asp,2006

[12] Li Ying, Qiu Ben, Cao Yiqun. A new DNS for network virus inspection and location[J]. Computer Engineering, 2005, 31(19): 56-58(in Chinese)

Author

Li Ying, female, was born in 1973, received her Doctor's degree in Tianjin University. She is an associate professor in School of Business, Sun Yat-Sen University and has published more than 20 papers in academic journals and conferences. Her research interests include network and complexity systems. She can be reached by E-mail: mnsliy@sysu.edu.cn

---

Foundation item: Project is supported by the National Nature Science Foundation of China (Grant No. 60672142, 60772053, 90304005), New architecture and technology for the Internet information sharing, The relation and Countermeasure of Application Layer Behavior and Collective Performance of Internet, Forecast of Traffic Model on Internet; Supported by National Basic Research Program of China, (Grant No. 2007CB307100, Project No. 2007CB307105), The Theory of Mobility and Sensor Network under The Integrated Network and the Universal Applicable Service System.