L-030

Design of User Support System for Combating against Malware

Nobutaka Kawaguchi[*1] Takayuki Yoda[*1] Tatsunoshin Kawaguchi[*1] Toshihiko Kasagi[*2] Yuji Hosizawa[*3]
[*1] Hitachi, Ltd. [*2] KDDI Corporation [*3] SecureBrain Corporation

Abstract-- In paper, we describe a design of a user support system for comprehensive countermeasures against malware by coordinating various anti-malware functions that are developed by different security organizations individually. As a lot of new malware appears every day, most of conventional anti-malware solutions relying on signature strings become less effective. In meantime, many security organizations develop their own anti-malware functions such as behavior analyzers. However, it is difficult for most end users with little knowledge on security to effectively utilize them, because each of the functions implements only a part of the procedure required for malware countermeasures, and the single use of each function does not achieve comprehensive solution. Our proposed system divides the procedure into several functions, and allows each security organizations to focus on developing particular functions individually, taking advantage of their expertise and strength. Then, the system coordinates the various anti-malware functions through standardized interfaces, and provides end users with a comprehensive countermeasure, ranging from suspicious file detection in user PCs to malware removal from user PCs.

1. Introduction

In these days, many new species and variants of malware are appearing everyday. Existing conventional anti-malware software, which is installed in users PCs and relies on signature strings, has become less effective because the rate of signature update can no longer keep up with a rate at which new malware appears. Moreover, many recent malware is equipped with functions of dynamically changing the appearance such as file structure to evade detection.

To counter against the malware, anti-malware web services that detect and remove malware from users PCs are emerging. In the services, suspicious files or their signatures in users PC are uploaded to the service cites and analyzed. If malware is detected through the analysis, the corresponding vaccine is generated and distributed to users. The services gather latest threat information in real-time and frequently update the large-scale signature databases in order to catch up with new kinds of malware immediately.

However, developing and running such services will impose significant costs and burdens on the providers, because they need to design, implement and integrate all the steps necessary to counter malware, ranging from suspicious file transmission between PCs and service cites to the generation of vaccine programs. Thus, we consider that only a few large venders with abundant resources can operate such services.

In the meantime, many security vendors and research institutions are developing their own original anti-malware functions such as behavior analyzers and whitelist databases, which we will describe in later sections. Some of the functions are promising approaches using the latest technologies, and can be more effective than the corresponding parts of the integrated services. However, since the single use of each function does not achieve a comprehensive solution against malware, it is not easy for average PC users without knowledge on malware or security to make use of the functions effectively.

We are now developing a system for malware countermeasure by coordinating various such anti-malware functions provided by different security organizations. The aim of this system is to provide PC users with a rapid and comprehensive solution against new malware with low cost compared to the densely integrated services. While each of the anti-malware functions is insufficient for the comprehensive countermeasure from suspicious files detection to malware removal, combination use of the functions will achieve the goal.

This system divides the procedure required for the comprehensive countermeasure into several functions such as malware analyzer and vaccine program generator, and allows each security organization to focus on developing particular functions individually, taking advantage of their expertise and strength. Finally, the functions are coordinated with others, and operate as a single service. This approach makes the system keep up with new trends of malware without high cost.

To coordinate the functions, this system defines standardized interfaces for each type of functions. Then, the functions are loosely bounded to the system coordinators through the interfaces.

As far as we know, this is the first approach that achieves a comprehensive countermeasure against malware by coordinating individually developed anti-malware functions.

The rest of paper is organized as follows. In Section 2, we discuss works related to malware countermeasures. In Section 3, we describe the design of our system and anti-malware functions. Finally, Section 4 concludes this paper and shows future works.

2. Related Works

2.1 Malware detection without signatures

As conventional anti-malware solutions using signature strings of malware becomes ineffective against recent malware, new approaches are now emerging. In this section, we introduce two attractive approaches that detect malware without malware signatures: behavior analysis and whitelisting.

Behavior analysis detects malware based on the behavior. There are a few platforms for behavior analysis such as Nicter [2] and others [1][3][6]. In the analysis, a target program is executed on controlled environments such as virtual machines, and how it behaves in the environments is monitored. For example,

- Files and registries the program accesses, creates, deletes and modifies,
- APIs the program calls
- Processes the program starts or stops
- Servers the program contacts

- Packets the program sends and receives

are recorded using API hooking and packet capturing. After the programs have been run for a few minutes, the platforms classify the program into either of malware or benign program based on the observations.

Whitelisting is another approach for malware detection. Whitelist is a list that includes hash values of known programs which are verified to be non-malicious. In whitelisting, programs not included in the list are classified as suspicious or malicious. This approach has an advantage that it rarely causes false negatives, and therefore is effective in detecting unknown malware. Now, there are some large whitelist databases available. For example, National Software Reference Library (NSRL) publishes a list of millions of MD5 and SHA-1 hashes computed from known benign programs [4].

Although the two approaches are essential to combat against new malware, neither achieves comprehensive countermeasures. In behavior analysis, users are required to select files to be analyzed, and manually upload them to the analysis platforms. In addition, when malware is detected, it is quite a difficult task for average users to completely revert changes on PCs caused by the malware, since malware usually create, modify and delete various files and registries.

On other hand, whitelisting has a critical drawback that benign programs that are not listed in the whitelist cause false positives. It is practically impossible to include hashes of all the existing benign programs completely in the whitelist.

2.2 Combination of anti-malware functions

There is quite a limited number of works on combining several anti-malware functions to achieve high-performance countermeasure. CloudAV [5] is a platform for integrating multiple malware detectors, including traditional AV tools and behavior analysis systems. In the system, target files are checked by various detectors, and the final judgments are done based on the aggregated results. However, this system only supports a part of countermeasure: automated file uploading from use PC to the platform, and detection of the files. Even if a file is deemed malware, no vaccine programs are provided to users. Moreover, since no standardized interface is defined, the operator needs to manually integrate each detector with the platform, which will be very cumbersome.

3. Proposed System

3.1 Concept

As mentioned in the Introduction, the proposed system coordinates various anti-malware functions, which are developed by different organizations, in order to achieve comprehensive countermeasure with low cost. Here, in our definition, comprehensive countermeasure includes the steps of automated suspicious file listing, malware analysis and detection, vaccine program generation and malware removal.

The most significant differences between our approach and existing ones are as follows:

- Proposed system conducts not only detection but comprehensive countermeasure against malware by using various types of anti-malware functions.
- By dividing the procedure for countermeasure into a few types of anti-malware functions and defining their standardized interfaces, this system easily interchange various functions that implement the interface. Also, developers can focus on only the particular part of this system in which they have expertise and strengths.

The comprehensiveness of countermeasure and division of labor of the development enable the proposed system to keep up with the advances of new malware.

The targets of this system range widely, from mobile users to enterprise users. In particular, we aim at providing countermeasures to not only high-end desktop PCs but also mobile devices such as smart phones that have quite limited CPU power and storage space. As mobile devices become complex, there is an increasing need for protecting mobile devices. Thus, anti-malware functions at user side should be lightweight and not consume many computation resources in mobile devices.

3.2 System Overview

Figure.1 shows the overview of this system. The system divides the countermeasure into four types of anti-malware functions: the suspicious file detector, the benign file filter, the malware analyzer and the vaccine program generator. Also, there are two types of coordinators: the client agent and the server agents installed in user PC and user support center respectively. The coordinators coordinate the functions and serve them as a single security service to PC users.

In the following sections, we describe how this system detects and then removes malware from user PCs, and details on roles and requirements of the four anti-malware functions.

3.3 Procedure of detection and removal

We describe the procedure of detection and removal of malware step by step.

1. First of all, a user downloads the client agent from the user support center. When executed, the client agent runs suspicious file detector routinely, and obtains a list of suspicious files at the PC.
2. The client agent transmits the suspicious files to the server agent, along with the environment information of the PC. Before transmission, the agent shows the user a list of suspicious files. Then, the user removes files that should not be transmitted from the list. After the user approves the list, the files are transmitted. For safe transmission, each file is encrypted with a key shared between the user support center and each client agent.
3. On receiving the suspicious files, the server agent decrypts the files and passes them to the benign file filter. The filter returns whether the files are known benign files or not. Then, the server agent sends list of known benign files to the client agent. The list is cached in the client agent, and the corresponding files will be never sent to the server agent again.
4. The server agent sends the remaining unknown files and the environment information to the

malware analyzer. Then, the malware analyzer outputs an analysis report including detection result for each file. The list of files classified as begin by the analyzer is cached in the server agent and the client agent.

5. The server agent sends files which are determined as malware, the analysis reports and the environment information to the vaccine program generator, and then obtains the vaccine programs.
6. The server agent sends the vaccines to the client agent. The client agent executes them on the user PC to remove malware. The results of the executions are fed back to the server agent if the user will. On the server agent side, vaccines that are proved to work appropriately are cached, and are later prescribed to users who send the same malware files.

As shown above, the server agent caches vaccine programs and files classified as begin. The caches will be significantly effective in reducing requests for anti-malware functions and, shorten the response time taken to prescribe vaccines to the users. For instance, since the regular update of operating systems will involve the installation of new unknown files, many users will send the same files at the update. Also, when malware pandemic happens, many users will need the same vaccine programs.

3.4 Suspicious File Detector

The suspicious file detector is a component installed in a user PC. The requirement for the detector is to scans files in the PC under the control of the client agent, and detects suspicious files while keeping almost zero false negatives. Note that the aim of this detector is not to detect malware, but rather to detect suspicious files. Suspicious files are executables that can possibly be malware. Using this detector, the proposed system determines candidate programs to be sent to malware analyzers and vaccine program generators. Thus, malware that the detector fails to detect will be never analyzed or removed from PCs. Therefore, quite small false negative rate is required for the detector. On the other hand, certain amounts of false positives are allowed in return, although this can cause burdens on systems and networks.

While we leave the implementation details for developers, one typical approach for achieving the requirements is to classify all executables that do not have digital signatures issued by trusted parties as suspicious. More sophisticated approaches may check features specific to malware such as very long file name and anti-debugging codes. Since scanning all files in PCs can impose high overload on PCs, full scans should be performed only when CPU usage rate is low enough.

Next, we briefly describe the interface between the detector and the client agent. The communication is done through directories and files. When the detector finds a suspicious file, it creates a file that includes the suspicious file path and the detection time. Then, the detector puts the file into a specified directory. The client agent monitors the directory and obtains the lists of suspicious files.

3.5 Benign file filter

The benign file filter is a filter that takes suspicious files as input, and returns whether the files are known benign files or not. The proposed system uses the filter to reduce the number of files that the malware analyzers need to analyze. Thus, the benign filter is positioned as a preprocessing step of malware detection. Whitelisting, which we discussed in Section.2, is a typical implementation of the filter.

The communication between the server agent and the benign file filer is done through SOAP messages. Currently, we define one operation, checkSample. checkSample is used to judge whether an input file is a known benign file. The response of this operation is either of true or false.

3.6 Malware Analyzer

The malware analyzer analyzes an input file, and returns whether the file is malware or not. Behavior analysis, which we discussed in Section 2, is a typical approach. Another approach is static analysis that inspects the structure of the file [8]. An important requirement for the malware analyzer is to output an analysis report, which is used to generate vaccine programs in later. While the format of analysis report is under formalization currently, it should include information on whether the analyzed file is malware or not, and changes caused by malware to the PC such as file/registry modification, creation and deletion. For example, reports should include entries like that:

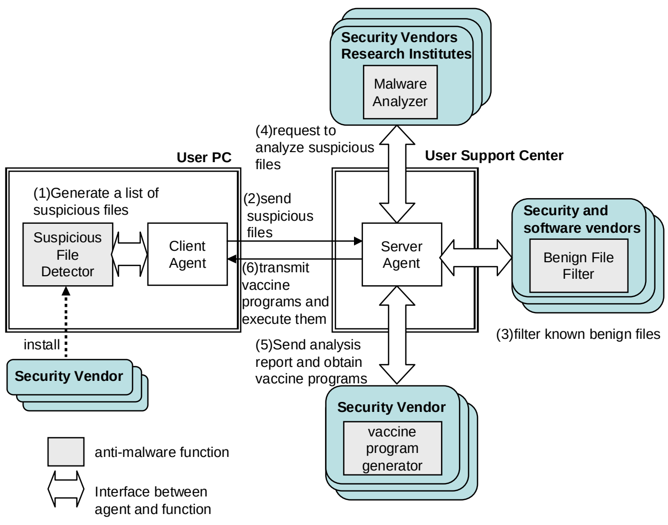
Security Vendors Research Institutes
Malware Analyzer
(4)request to analyze suspicious files
User PC
User Support Center
(1)Generate a list of suspicious files
Suspicious File Detector
Client Agent
(2)send suspicious files
(6)transmit vaccine programs and execute them
Server Agent
Security and software vendors
Benign File Filter
install
Security Vendor
(5)Send analysis report and obtain vaccine programs
(3)filter known benign files
Security Vendor
vaccine program generator
anti-malware function
Interface between agent and function

Figure 1 System Overview

```
<ANALYSIS>
    <FILE_NAME> sample.exe </FILE_NAME>
    <SHA-1 HASH> 123456789abc…</SHA-1 HASH>
    <CLASSIFICATION> Malware </CLASSIFICATION>
    <BEHAVIOR>
        <CREATE_FILE> C:\malware.exe </CREATE_FILE>
        <MODIFY_FILE> C:\WINDOWS </MODIFY_FILE>
    </BEHAVIOR>
</ANALYSIS>
```

Since the behavior of malware could depend on the execution environments, the malware analyzer may need environment information on the user PC that submits the file (e.g. versions and settings of operating system and installed applications) for accurate analysis. While most existing analyzers do not use this information, we handle this information as an option of the input interfaces.

The communication between the server agent and the malware analyzer is done through SOAP messages. We currently define three types of operations; registerSample, getAnalyzedID, and getReport. registerSample is used to register a file to be analyzed. The response of this operation is a register ID. The server agent checks whether the analysis is completed by getAnalyzedID. This operation takes register IDs as input, and return register IDs of files that the analyzer has already analyzed. Finally, reports are retrieved by calling getReport with a register ID as input.

3.7 Vaccine Program Generator

The vaccine program generator generates vaccine programs that remove malware and revert PCs back to the original states. The generator is required to be equipped with two functionalities: vaccine generation based on analysis reports, and verification of the generated vaccine programs.

To generate a vaccine program for a malware, the generator uses the analysis reports to obtain information on changes caused by the malware on PCs. Based on the information, how the vaccine program recovers a compromised PC is determined. Especially, recovering files/registries that are modified or deleted is a challenging task. According to [7], only about 75% of primary files (e.g. .exe, .bat, .dll) and 4% of the remaining files can be reverted by using existing anti-malware tools.

After generating a vaccine, the generator verifies that it removes the malware and related data properly without causing harmful effects to PCs. For verification, the target malware, and then the vaccine program are executed on a test environment that is configured with environment information of the PC submitting the malware, and then whether the environment is properly reverted to the original state or not is checked. To certainly remove malware that exhibit different behavior each time it is executed, through verification is required. After verification is completed, the vaccine program is passed to the server agent.

Currently, the interface between the server agent and the vaccine program generator is under formalization

4. Conclusion and Future works

In paper we have described a design of a user support system for comprehensive countermeasures against malware by coordinating four types of anti-malware functions developed by various security organizations individually.

This system differs from existing approaches in that this system achieves not only detection but comprehensive countermeasure against malware, and that the required procedures are divided into independent functions with standardized interfaces.

In future works, we will fix the requirements and interfaces of anti-malware functions, and implement the reference model with Nicter as a malware analyzer. Then, we will evaluate the performance of this system through experiments.

ACKNOWLEDGEMENT

This research is supported by the National Institute of Information and Communications Technology, Japan.

REFERENCES

[1] C.Willems, et al.,“Toward Automated Dynamic Malware Analysis Using CWSandbox”, IEEE Security and Privacy Magazine, Vol.5, Issue 2, 2007.

[2] D.Inoue, et al.,“An Incident Analysis System nicter and Its Analysis Engines Based on Data Mining Techniques”, Proc. of ICONIP2008, 2008.

[3] Anubis: Analyzing Unknown Binaries, http://anubis.iseclab.org/, accessed at 02/16/2010.

[4] National Software Reference Library, http://nsrl.nist.gov/, accessed at 02/16/2010.

[5] Jon Oberheide, et al., CloudAV: N-Version Antivirus in the Network Cloud, In Proc. of the 17th Usenix Security Symposium, July, 2008.

[6] Norman Solutions. Normand sandbox whitepaper, http://download.norman.no/whitepapers/whitepaper_Norman_SandBox.pdf, 2003.

[7] Emanuele Passeriniy, et al., “How good are malware detectors at remediating infected systems?”, In Proc. of DIMVA’09, 2009.

[8] Christopher Kruegel, et al., “Polymorphic worm detection using structural information of executables”, In Proc. of RAID’05 ,2005.