Anti-Malware Technologies

Mitsutaka Itoh†, Takeo Hariu, Naoto Tanimoto, Makoto Iwamura, Takeshi Yagi, Yuhei Kawakoya, Kazufumi Aoki, Mitsuaki Akiyama, and Shinta Nakayama

Abstract

This article introduces anti-malware technologies currently being researched and developed at NTT. Malware is used by cyber attackers to abuse users' personal computers and servers, such as using them as bots, which exposes computer networks to security threats. Our anti-malware technologies can detect malware infection activities, collect malware, analyze infection routes, and analyze the malware itself. They will lead to a safe and secure network in which malware is filtered out.

1. Introduction

With the widespread use of the Internet, the number of security problems has also increased. Cyber crime such as identity theft has dramatically increased in recent years. Most of the security problems are caused by malware, which is programmed and used by attackers. By operating malware on users' personal computers and servers, attackers can control those personal computers and servers illegally as bots (programs designed to automate tasks) to engage in other cyber attacks.

Malware infection activities take advantage of vulnerabilities not only in the Windows operating system (OS) but also in web applications on servers and in web browsers on personal computers. Such activities are become increasingly complex so it is difficult to detect them and identify infection routes. Furthermore, the variations of malware are increasing rapidly, making malware analysis all the more difficult. For example, a downloader (one type of malware) does not infect a victim directly but causes the victim to download other malware programs that do cause infection.

At NTT Information Sharing Platform Laboratories, we are researching and developing technologies for detecting malware infection activities, collecting malware, analyzing infection routes, and analyzing malware itself. These technologies let us identify malware infection characteristics accurately and efficiently. Consequently, a safe and secure network will be achieved by filtering malware.

† NTT Information Sharing Platform Laboratories
Musashino-shi, 180-8585 Japan

2. Overview of anti-malware technologies

The anti-malware technologies that we have developed are divided into three main categories, as shown in **Fig. 1**. First, we introduce technologies for detecting malware infections and collecting malware; then, we explain malware analysis technologies.

2.1 Malware infection detection and malware collection

These technologies are used to receive attacks and collect malware through the use of decoy systems called *honeypots*. They are also used to analyze communications with honeypots and identify information that can be useful in defending against malware infections.

Honeypots can generally be classified into low-interaction and high-interaction types. Low-interaction honeypots are designed to emulate vulnerable systems in order to attract malware. They can collect attack information safely because their systems do not execute the malware. However, the attack information acquired is limited because attackers cannot compromise such systems as they are just interacting

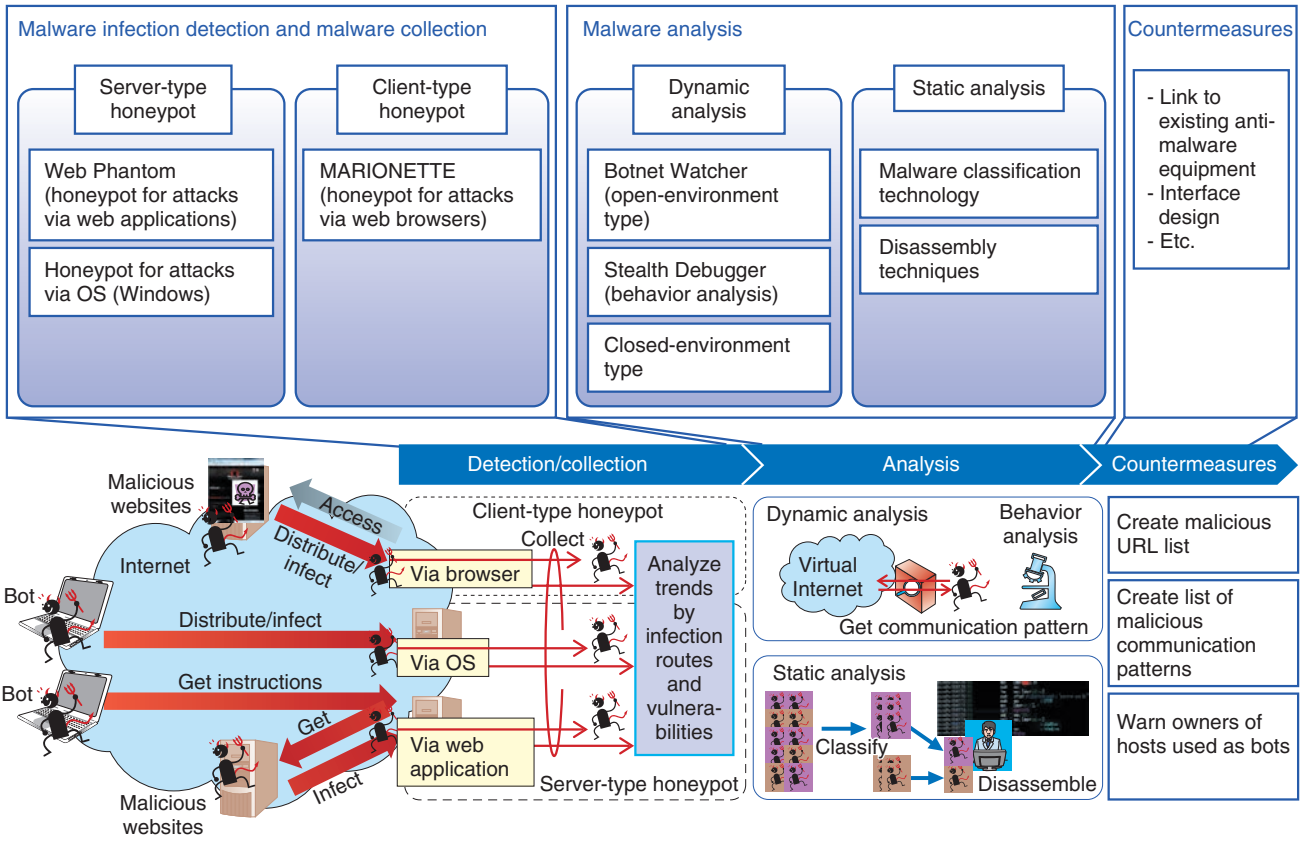


Fig. 1. Overview of anti-malware technologies.

with a simulation. On the other hand, high-interaction honeypots are composed of real vulnerable systems, which can actually be attacked through these vulnerabilities, with embedded surveillance functions. High-interaction honeypots are difficult to manage because their systems temporarily execute the attacking code, but they can collect much more attack information than low-interaction honeypots. Although the risk of high-interaction honeypots being infected with malware is said to be a problem, we have succeeded in developing secure high-interaction honeypots. We are researching and developing three kinds of honeypots for different types of major malware-infection activities. The first type can detect activities that exploit the vulnerability of the Windows OS. The second type (web server honeypot) can detect activities that exploit vulnerabilities in web applications. The third type (web client honeypot) can detect activities that exploit vulnerabilities in web browsers.

2.2 Malware analysis

These technologies analyze malware collected by honeypots and analyze its functions in detail to clarify potential threats. Malware analysis consists of dynamic and static types: dynamic analysis actually runs malware to analyze its behavior while static analysis analyzes the malware program code.

Malware dynamic analysis can be classified into closed- and open-environment types: closed-environment analysis runs malware in an environment completely isolated from the Internet while open-environment analysis runs it in one connected to the Internet. In both environment types, a debugger is useful for monitoring the behavior of malware in detail. The communication patterns of malware obtained by dynamic analysis can be used to determine whether there are any users' personal computers or servers infected by malware.

Malware static analysis uses disassembly techniques and other means to clarify all of the functions possessed by a certain piece of malware. To make this process efficient, malware is classified into groups

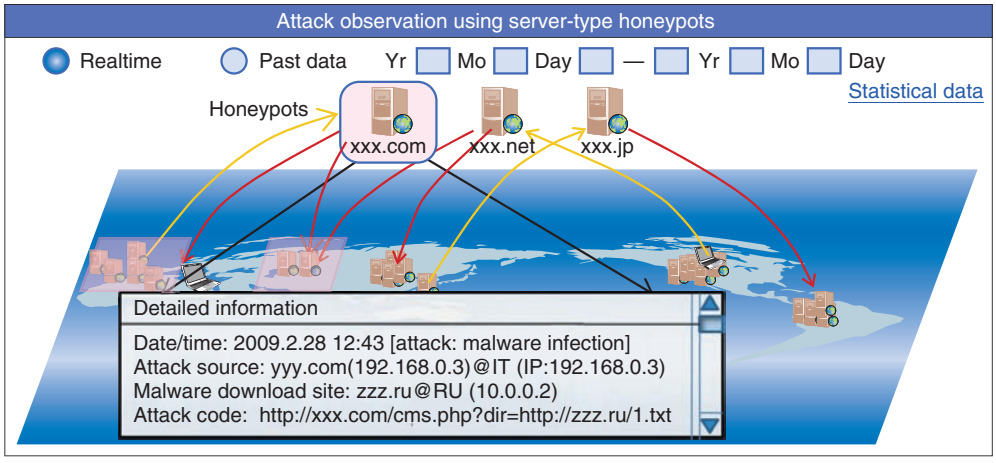


Fig. 2. Screenshot of malware infection activities monitored by Web Phantoms.

based on program similarities, and only malware representative of each group is analyzed.

3. Honeypots

3.1 Web server honeypot

With the increasing popularity of weblogs (blogs) and online shopping, many web applications provide a platform for social services. The increase in the number of web applications has made it easy for attackers to hack websites by using exploit code targeting the vulnerabilities of those web applications. For example, attackers can illegally obtain and control information stored on websites. Furthermore, they can even infect websites with malware. To defend against such attacks, web application firewalls and log monitors are deployed. However, to prevent false-negative and false-positive detections, it is necessary to customize the settings according to the configuration of each website. To solve this problem, we are researching and developing a high-interaction web honeypot called Web Phantom, which collects attacks on websites from the Internet and automatically identifies information that can be used to detect and filter website attacks.

The structure of Web Phantom makes it easy to install vulnerable web applications by using normal procedures. It also ensures safety with respect to the risk of malware infection by monitoring the execution of attack code. When the system detects the end of the execution, it returns to the state it was in prior to accepting the attack code. This function enables the Web Phantom to collect information about sequences of attack operations caused by attack code, such as malware downloaded from the Internet, safely and automatically. In addition, the system recreates attacks in a closed environment by using the collected information in order to precisely identify communications that cause an illegal state such as information falsification or malware infection. For example, when the system detects a malware infection, the exploited web application vulnerability can be identified and the URL (uniform resource locator) of the malware download site on which the attacker placed the malware, can be determined. Moreover, a centralized monitoring system that can collect information from many Web Phantoms has been constructed (**Fig. 2**). Using this monitoring system, we can gather a significant amount of useful information efficiently. By deploying this system, we can discover new vulnerabilities in web applications. Moreover, user websites are protected from malware infection by filtering communications from the websites to malware download sites.

3.2 Web client honeypot

Nowadays, the number of web-browser-targeting attacks that lead users to adversaries' websites and exploit web browser vulnerabilities is increasing. To comprehend the exploitation techniques and provide effective countermeasures, we have designed and implemented a high-interaction client honeypot, called MARIONETTE, that detects web browser exploitation in accurate detail.

The key features of MARIONETTE are stepwise detection focusing on exploitation phases, multiple

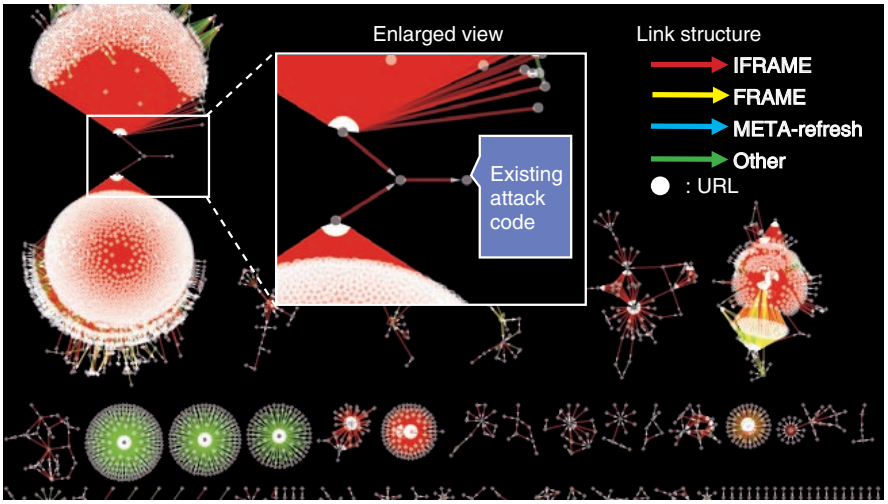


Fig. 3. Visualization of malware distribution network targeting web browsers.

crawlers, malware infection prevention, and malware distribution network tracking. HoneyPatch, which is one of the detection methods, monitors the dataflow of certain vulnerable functions in web browsers and decides whether a vulnerable point is being attacked. Process Sandbox also detects malicious behavior and prevents infection in order to maintain continuous system operation. In addition, by running multiple crawler processes, MARIONETTE can discover malicious sites in a huge web space.

In most situations, an adversary takes the form of a redirection network of malicious sites (i.e., exploit, hopping, and malware distribution sites). An exploit site does the actual exploitation of a target web browser and forces it to download executable malware from a malware distribution site. A hopping site redirects a target to the next hopping or exploit site. Adversaries lure web clients to exploit websites by using compromised websites that have been injected with malicious redirection code. Once web clients access these compromised websites, they are automatically redirected to the next hopping or exploit site. We call the redirection network formed by these sites a malware distribution network. MARIONETTE has a function for extracting the structure of malware distribution networks.

Our lengthy investigation reveals the nature of a malware distribution network, which has a vast number of hopping sites embedded in the redirection code toward a specific exploit site (**Fig. 3**). The information obtained by MARIONETTE can be used to provide countermeasures appropriate to the situation. For example, the owner of a tampered hopping site can be notified of requesting to correct the tampered web content and we can filter the accesses to an exploit site embedded in exploit code.

4. Malware analysis technologies

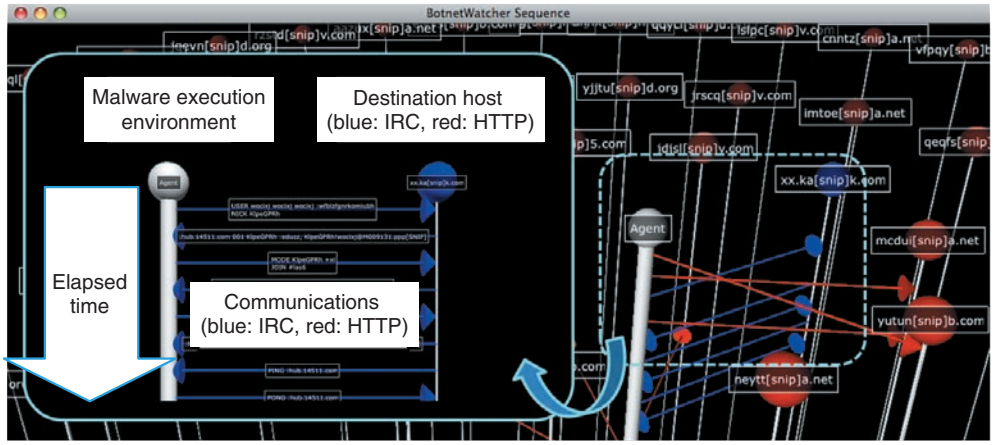
4.1 Dynamic analysis

The dynamic analysis of malware can take place in two types of execution environment: an environment that is isolated from the Internet or one that is not. Since some existing malware requires access to the Internet (e.g., bots and downloaders), dynamic analysis in an isolated environment cannot obtain a full set of results. However, executing malware in a non-isolated environment is difficult because there are some concerns about how to control the activities of malware that has access to the Internet.

In addition, malware analysts always use a debugger to obtain detailed analysis results of malware, but some sophisticated malware can detect debugger activities and stop its own execution. So if we want to analyze malware using a debugger, we must hide the debugger's activities.

In order to analyze malware safely, we have constructed Botnet Watcher, which executes malware in a non-isolated environment. We have also constructed Stealth Debugger, which implements debugging mechanisms in a virtual machine monitor (VMM) layer.

When malware tries to connect to the Internet, Botnet Watcher terminates the connection and inspects



HTTP: hypertext transfer protocol
IRC: Internet relay chat

Fig. 4. Visualization of malware activities on the Internet.

the payload. If the inspection reveals communication between the bot and a command and control (C&C) server or the downloading of other malware, Botnet Watcher transfers these communications to the Internet. However, if the malware begins to send malicious payloads (e.g., DDoS attacks or spam email), these are sent to a virtual Internet, which creates pseudo-responses to the malware. In this way, Botnet Watcher can safely obtain the features of the communication patterns between the malware and other hosts (**Fig. 4**).

Stealth Debugger, which is implemented as a debugging mechanism in the VMM layer, provides a debugging function from another layer of the malware execution environment. Because ordinary debuggers make use of CPU (central processing unit) or OS debug-support mechanisms, some kinds of malware can easily detect the operation of a debugger. In contrast, Stealth Debugger provides the debugging function from the VMM layer independently of the original debug-support mechanisms, so malware cannot detect any debugger activities. Thus, Stealth Debugger can get effective malware analysis results by means of some anti-analysis techniques and can elucidate malware functions quickly.

4.2 Static analysis

The number of malware programs has been steadily increasing in recent years, and while it would be unrealistic to come up with a countermeasure for each and every one, determining a priority for creating countermeasures is proving to be difficult. Under these circumstances, we have been researching malware static analysis technologies centered on the classification of malware with the aim of clarifying the overall content of malware. Our malware classification technology calculates the similarity between malware programs by examining the behavior of malware rather than its program code. This approach enables malware programs to be classified according to their potential functions such as ones that are not executed without a command from the attacker. However, much of the recent malware conceals the original program code through the use of an obfuscation tool. It is also hard to obtain information about malware symbols, which makes it difficult to disassemble binary streams mixed with machine-language instructions and data. In response to these problems, we have developed three malware analysis techniques and constructed a malware classification system by combining them. These are (1) an unpacking technique that extracts dynamically generated code, (2) a technique that computes the most likely disassembly result based on a probabilistic model, and (3) a technique that efficiently computes the longest common subsequence of machine-language instruction sequences. The results of clustering 3232 samples of malware that we collected using this system are shown in **Fig. 5**. The names of these 3232 samples are lined up along the periphery of the circle. Samples connected by arcs with large radii exhibit high similarity while those connected with arcs near the center

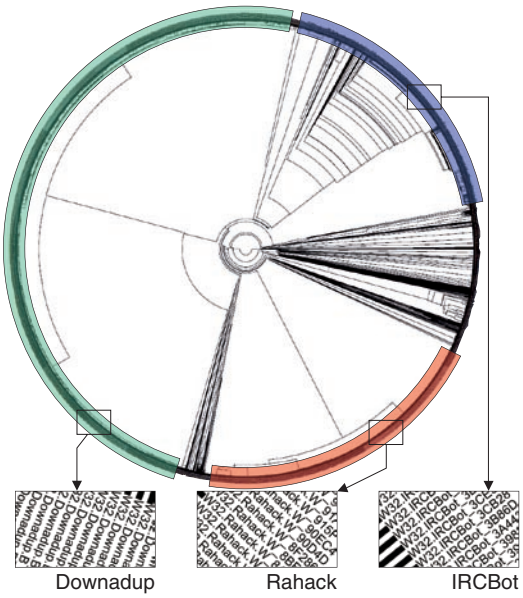


Fig. 5. Results of classifying 3232 malware samples.

of the circle exhibit low similarity. As a result of this classification, we found that about 50% of all samples were made up of Downadup malware, about 25% of Rahack malware, and about 20% of IRCBot malware (IRC: Internet relay chat). These results show that, for the malware samples targeted by this classification, most malware functions could be determined by analyzing representative samples of only a few types without having to analyze all samples.

5. Conclusion

Anti-malware technologies that we have developed can identify malicious URLs, which should not be accessed from personal computers and servers, in a form that can be input to existing filtering equipment. Furthermore, they can also identify infected user equipment with high accuracy based on the features of malware communications and functions. In the near future, we plan to evaluate these technologies by performing tests with prototype systems.

Mitsutaka Itoh
Senior Research Engineer, Supervisor, Secure Communication Project, NTT Information Sharing Platform Laboratories.
He received the B.S. and M.S. degrees in mathematics from Waseda University, Tokyo, in 1984. He joined NTT Laboratories in 1984. Since then, he has been engaged in network security R&D. He created NTT-CERT and has contributed to improvements in information-system security. His current research interests are VoIP security, mobile cloud computing, web security, and countermeasures against botnets. He is a member of the Institute of Electronics, Information and Communication Engineers (IEICE) of Japan and the Information Processing Society of Japan.

Takeo Hariu
Senior Research Engineer, Supervisor, Secure Communication Project, NTT Information Sharing Platform Laboratories.
He received the M.S. degree in electro-communications from the University of Electro-Communications, Tokyo, in 1991. Since joining NTT in 1991, he has been engaged in network security R&D. He is a member of IEICE.

Naoto Tanimoto
Research Engineer, Secure Communication Project, NTT Information Sharing Platform Laboratories.
He received the B.E. degree in engineering from the University of Electro-Communications, Tokyo, in 1989. Since joining NTT in 1989, he has been engaged in operating systems R&D.

Makoto Iwamura
Research Engineer, Secure Communication Project, NTT Information Sharing Platform Laboratories.
He received the M.E. degree in science and engineering from Waseda University, Tokyo, in 2002. He joined NTT in 2002. He is currently working toward the D.E. degree at the Graduate School of Fundamental Science and Engineering, Waseda University. His research interests include reverse engineering, vulnerability findings, and malware analysis.

Takeshi Yagi
Secure Communication Project, NTT Information Sharing Platform Laboratories.
He received the B.E. degree in electrical and electronic engineering and the M.E. degree in science and technology from Chiba University in 2000 and 2002, respectively. Since joining NTT in 2002, he has been engaged in network architecture R&D. His current research interests include network security. He is a member of IEICE and the Institute of Electrical Engineers of Japan.

Yuhei Kawakoya
Secure Communication Project, NTT Information Sharing Platform Laboratories.
He received the M.E. degree in science and engineering from Waseda University, Tokyo, in 2002. Since joining NTT in 2005, he has been engaged in network security R&D.

Kazuhumi Aoki
Secure Communication Project, NTT Information Sharing Platform Laboratories.
He received the M.S. degree in information science from Tohoku University, Miyagi, in 2006. Since joining NTT in 2006, he has been engaged in network security R&D.

Mitsuaki Akiyama
Secure Communication Project, NTT Information Sharing Platform Laboratories.
He received the M.E. degree in information science from Nara Institute of Science and Technology in 2007. Since joining NTT in 2007, he has been engaged in network security R&D.

Shinta Nakayama
Secure Communication Project, NTT Information Sharing Platform Laboratories.
He received the M.S. degree in engineering from the University of Electro-Communications, Tokyo, in 2009. Since joining NTT in 2009, he has been engaged in network security R&D.