Journal of Information Processing Vol.20 No.4 835–845 (Oct. 2012)

[DOI: 10.2197/ipsjjip.20.835]

Regular Paper

Malware Sandbox Analysis with Efficient Observation of Herder's Behavior

Takahiro Kasama[1,2,a)] Katsunari Yoshioka[1,b)] Tsutomu Matsumoto[1,c)]
Masaya Yamagata[3,d)] Masashi Eto[2,e)] Daisuke Inoue[2,f)] Koji Nakao[2,g)]

Received: December 1, 2011, Accepted: June 1, 2012

Abstract: Recent malware communicate with remote hosts in the Internet for receiving C&C commands and updating themselves, etc., and their behaviors can be diverse depending on the behaviors of the remote hosts. Thus, when analyzing these malware by sandbox analysis, it is important not only to focus behaviors of a malware sample itself but also those of the remote servers that are controlled by attackers. A simple solution to achieve this is to observe the live sample by an Internet-connected sandbox for a long period of time. However, since we do not know when these servers will send meaningful responses, we need to keep the sample being executed in the sandbox, which is indeed a costly operation. Also, leaving the live malware in the Internet-connected sandbox increases the risk that its attacks spill out of the sandbox and induce secondary infections. In this paper, we propose a novel sandbox analysis method using a dummy client, an automatically generated lightweight script to interact with the remote servers instead of the malware sample itself. In the proposed method, at first we execute a malware sample in the sandbox that is connected to the real Internet and Internet Emulator. Secondly, we inspect the traffic observed in the sandbox and filter out high-risk communications. The rest of the traffic data is then used by the dummy client to interact with the remote servers instead of the sample itself and effectively collects the responses from the servers. The collected server responses are then fed back to the Internet Emulator in the sandbox and will be used for improving observability of malware sandbox analysis. In the experiment with malware samples captured in the wild, we indeed observed a considerable number of changes in the responses from the remote servers that were obtained by our dummy client. Also, in comparison with the simple Internet-connected sandbox, the proposed sandbox could improve observability of malware sandbox analysis.

Keywords: malware, sandbox analysis, dummy client

1. Introduction

As the Internet has become essential medium in our life, security threats (e.g., divulging of private information, phishing, and denial of service attack (DoS)) have also increased. In these security threats, malware, which is a generic term for computer viruses, worms, Trojan houses, spywares, adwares, and bots, plays a significant role. Consequently, great research efforts have been taken to tackle them. With the recent explosive increase of malware, it is becoming nearly impossible to analyze them all by manual analysis using the reverse engineering techniques. As an effective countermeasure for this problem, malware sandbox analysis [2], [3], [5], [12], [14], [15], [16], [17], [18], [19], [20], [23], [25] has been studied widely. Its basic idea is to actually execute a captured malware sample in a testing environment (i.e., sandbox) to observe and analyze its behaviors. One of the advantages of the sandbox analysis is that it is not disturbed by packing and code obfuscation techniques, which are often used by malware developers to make static reverse engineering more time-consuming. Another advantage is that the sandbox analyzer can be implemented in highly automated fashion. However, there are several important issues to be addressed in malware sandbox analysis. One of them is that it can indeed observe only a single execution path of the malware sample by each execution and important behavior which can be crucial for developing an efficient countermeasure may not be observed. Especially, because recent malware communicate with remote hosts in the Internet for receiving C&C commands and updating themselves, etc., their behaviors can be diverse depending on the behaviors of remote hosts. In fact, the paper [16] reports that the behavior of malware observed by malware sandbox analysis can differ greatly when the analyses are done in two different time periods because some of the remote servers which the analyzed samples communicated with changed their behaviors over time. Therefore, in malware sandbox analysis, it is important not only to focus behaviors of malware sample itself but also those of the remote servers that are controlled by attackers. A simple solution to achieve this is to observe the live sample by an Internet-connected sandbox for a long period of time. However, since we do not know when these

[1] Yokohama National University, Yokohama, Kanagawa 240–8501, Japan
[2] National Institute of Information and Communications Technology, Koganei, Tokyo 184–8795, Japan
[3] NEC Corporation, Minato, Tokyo 108–8001, Japan
[a)] kasama@nict.go.jp
[b)] yoshioka@ynu.ac.jp
[c)] tsutomu@ynu.ac.jp
[d)] yamagata@da.jp.nec.com
[e)] eto@nict.go.jp
[f)] dai@nict.go.jp
[g)] ko-nakao@nict.go.jp

servers will send meaningful responses, we need to keep the sample being executed in the sandbox, which is indeed a costly operation. Also, leaving the live malware in the Internet-connected sandbox increases the risk that its attacks spill out of the sandbox and induce secondary infections.

In this paper, we propose a novel sandbox analysis method using a dummy client, an automatically generated lightweight script to interact with the remote servers instead of the actual sample. In the proposed method, at first we execute a malware sample in the sandbox that is connected to the real Internet and Internet Emulator. Secondly, we inspect the traffic observed in the sandbox and filter out high-risk communications. The rest of the traffic data is then used by the dummy client to interact with the remote servers instead of the sample itself and effectively collects the responses from the servers. The collected server responses are then fed back to the Internet Emulator in the sandbox and will be used for improving observability of malware sandbox analysis. For example, when next time a malware sample is analyzed and the remote servers cannot be connected, then, the dummy server can emulate the remote servers by sending the collected responses. As another example, when a new response from the remote servers is observed, we can re-analyze the malware sample which accesses the corresponding server and the response are fed back to the sample in order to observe its corresponding reactions. The advantage of the proposed method is that we can increase observability of the malware behavior by continuously monitoring many remote servers in parallel while not decreasing the efficiency too greatly by using light-weighted dummy clients instead of observing the interactions by live malware themselves. Besides that, since we can closely inspect the malware traffic and filter out potentially dangerous traffic before replaying it by the dummy client, containment of the outgoing attacks is also expected to improve.

We evaluated the proposed method with samples captured in the wild. By using a low risk containment policy of "emulating only the harmless HTTP and IRC connections," we were able to observe the changes of the server behavior and corresponding malware behavior. Also, in comparison with the simple Internet connected sandbox, the proposed sandbox could improve observability of the malware sandbox analysis, and revealed more behavior of malware.

The rest of this paper is organized as follows: In Section 2, we describe related works. In Section 3, we first explain several important properties of malware sandbox analysis and then explain the proposed sandbox analysis method. In Section 4, we explain the experiments for the evaluation of the proposed method. In Section 5, we discuss the challenge of proposed method. In Section 6, we give conclusions and future works.

2. Related Works

Malware sandbox analysis has been studied intensively in recent years. The previous studies of the malware sandbox analysis can be categorized into two approaches in terms of their Internet connectivity. One is a totally isolated sandbox and the other is a sandbox with the Internet connection. Examples of the former approach are Norman Sandbox [25], and [12], [14]. The limitation of this approach is that it is difficult to emulate remote hosts in the real Internet as malware make various kinds of communications. Especially, when they talk to one of their servers, such as C&C server and file servers, they can use arbitrary (even customized) protocols for data transmission and authentication, which makes the emulation increasingly challenging. The other approach is to carefully connect the sandbox with the real Internet. Examples of Internet-connected sandbox are CWSandbox [18], [20], Anubis [3], [19], Joebox [23], and [2], [16], [17].

The above literatures mainly focus on how to observe behaviors of a malware sample itself and therefore do not deeply discuss the issue of how the sample is influenced by the variety of responses from the remote servers such as C&C servers and malware download servers which are controlled by attackers. When viewed from this perspective, the analysis with totally isolated sandbox is not desirable because it cannot indeed observe the behavior of the remote servers in the Internet. By contrast, the analysis with the Internet-connected sandbox can observe how malware sample interact and get influenced by the remote servers. However, since we do not know when these servers send meaningful responses to the sample executed in the sandbox, we need to continue executing it to observe these responses. Other related technology aiming at a similar goal is to explore multiple execution paths. In Ref. [14], multiple execution paths of the malware sample can be observed by controlling the conditional branches. However, even exploring multiple execution paths cannot reveal additional functionalities of malware that are added by the interaction with the remote servers. In this paper, we propose a sandbox analysis using dummy client for solving this problem.

There are some related works which use a dummy client. Caballero et al. [5] performed a measurement study of the pay-per-install (PPI) market by infiltrating four PPI services in order to gather and classify the resulting malware executable files distributed by the services. They build and use a dummy client called "milker" in order to milk programs that the PPI services distributed. Their approach leverages techniques for automatic binary reuse [6], [13] which extract specific function defined by an analyst from an executable. Cho et al. [9] proposed a technique to infer complete protocol state machines and applied it to the analysis of botnet C&C protocols. They build a bot emulator which interacts with the C&C servers, reverse-engineer the message formats and their semantic content using automatic protocol reverse engineering [7], and extract encryption/decryption modules from the bot binary [6].

3. Malware Sandbox Analysis using Dummy Client

3.1 Properties

First, we show three important properties of malware sandbox analysis [17].

- Observability

Observability is a property of malware sandbox analysis in terms of observing malware behaviors in consideration. For those who are writing removal tools or AV signature may focus on the internal behaviors such as changes of registry keys and creation and deletion of files. Network administrator may be interested in

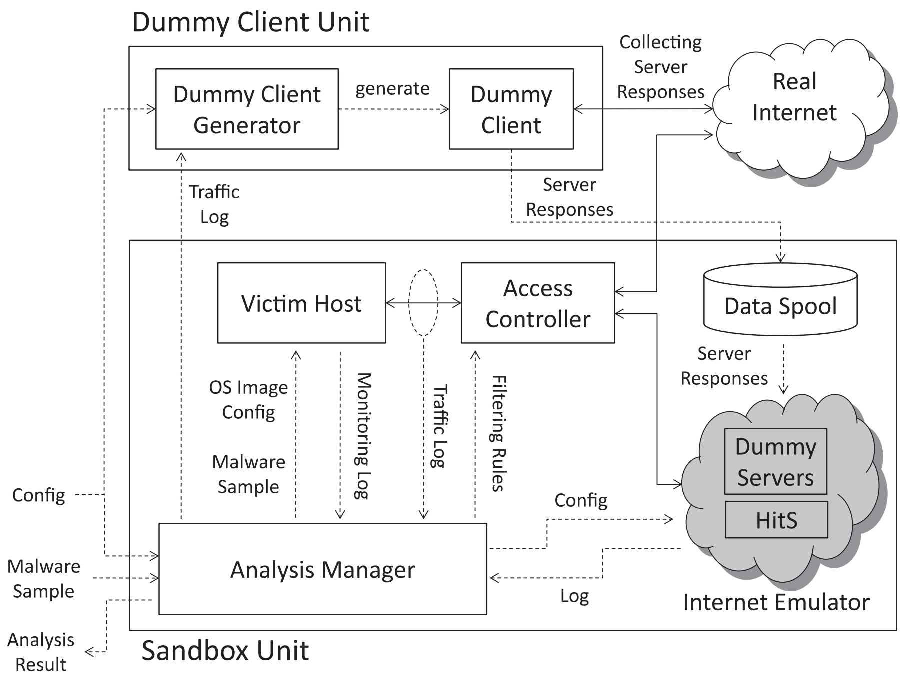

Fig. 1 Overview of proposed system.

their network behavior for writing an IDS signature. In any way, the sandbox analysis should be able to provide sufficient information to the analyst.

- Containment

Containment represents two sub-properties. One is the property for preventing the executed sample from attacking or infecting a remote host outside the sandbox. We term it as containment of outgoing attacks. The other is the property for suppressing a leakage of important information of the analysis system itself as they can be used against the system (i.e., sandbox detection). We term it as containment of system information.

- Efficiency

Efficiency is a property of malware sandbox analysis in terms of constantly providing analysis result with sufficient information in reasonable time.

3.2 Proposed Method

In this section, we explain our proposal, namely a sandbox analysis method using a dummy client. In our method, at first we execute a malware sample in the sandbox that is connected to the real Internet and the Internet Emulator. The Internet Emulator consists of many dummy servers and hosts with emulated/real vulnerable services, called Honeypots in the Sandbox (HitS). Secondly, we inspect the traffic observed in the sandbox and filter out high-risk communications such as port scanning, remote exploitations, and DoS. The rest of the traffic data is then used by the dummy client to interact with the remote servers. Then the dummy client continually interacts with the remote servers instead of the sample itself and effectively collects the responses from the servers. The accumulated server responses are fed back to the Internet Emulator in the sandbox and will be used for improving observability of malware sandbox analysis. For example, when next time a malware sample is analyzed and the remote servers cannot be connected, the dummy server can emulate the remote servers by sending the collected responses. As another example, when a new response from the remote servers is observed, we can re-analyze the malware sample which accesses the corresponding server and the response are fed back to the sample in order to observe its corresponding reactions. The advantage of the proposed method is that we can increase observability of the malware behavior by continuously monitoring many remote servers in parallel while not decreasing the efficiency too greatly by using light-weighted dummy clients instead of observing the interactions by live malware themselves. Besides that, since we can closely inspect the malware traffic and filter out potentially dangerous traffic before replaying it by the dummy client, containment of the outgoing attacks is also expected to improve.

First, we show the overview of the proposed sandbox method in **Fig. 1**. In the figure, the solid arrows indicate the communications by the analyzed malware sample and the dummy client, and the dotted arrows indicate communications by the sandbox system for its operation. The proposed sandbox consists of two units: Sandbox Unit and Dummy Client Unit. Sandbox Unit is where malware sample is actually executed and analyzed. This unit consists of five components, namely, victim host, Internet Emulator, access controller, analysis manager, and data spool. Dummy Client Unit is where the dummy clients are generated and executed. This unit contains two components: dummy client generator and dummy clients. Each component is described in Section 3.3. Also, the implementation of each component is described in Section 3.5.1.

3.3 Components

- Victim Host

The victim host is a host on which a malware sample is firstly executed to be observed. It is important that the security of the victim host is properly configured so that the executed malware makes further actions for us to observe.

- Access Controller

The access controller controls the traffic from the victim host. It receives all packets from the victim host and redirects them to either the Internet Emulator or the real Internet according to the filtering rules. The filtering rules are generated by the analysis manager.

- Internet Emulator

The Internet Emulator provides various network services to the victim host. The Internet Emulator consists of various dummy servers such as HTTP, SMTP, FTP, NTP, IRC, DNS, and Feedback servers. Feedback Server is the server to emulate behaviors of remote servers which malware interacts by sending the collected server responses. Beside those servers, it also deploys hosts with unpatched vulnerable services called a Honeypot-in-the-Sandbox (HitS). Suspicious traffic from the malware sample can be tested with HitS to see if it actually compromises it or not.

- Analysis Manager

The analysis manager is the core component that manages the entire analysis procedures. Based on a simple config file, it loads and refreshes OS image of the victim host, boots up and shuts down the victim host, executes malware sample in the victim host, receives and inspects all traffic logs and internal logs, sends all traffic logs to the dummy client generator, and finally outputs the analysis results to the analyst.

- Dummy Client Generator

The dummy client generator receives traffic logs from the analysis manager, inspects them, and generates a dummy client.

- Dummy Client

The dummy client is executed in the environments with the real Internet connection and emulates malware communications. When the dummy client received server responses, the responses are stored in the data spool.

- Data Spool

The data spool is the component which stores server responses collected by the dummy clients. The stored server responses are loaded by the Internet Emulator and sent back to malware executed in the victim host from dummy servers.

3.4 Analysis Procedure

The following is the brief procedure of the proposed sandbox analysis.

(1) The analyst inputs a malware sample and the configurations (OS image of the victim host, filtering rules for access controller, etc.) to the system.
(2) The analysis manager reflects the configurations to the access controller and Internet Emulator and boots up the victim host.
(3) The victim host executes the malware sample. All traffic from the victim host is first sent to the access controller, and the access controller redirects the traffic to either the real Internet or the Internet Emulator according to the filtering rules.
(4) After certain instructed time has passed, the victim host sends all traffic logs and internal monitoring logs to the analysis manager. Also, the Internet Emulator sends its logs to the analysis manager.
(5) The analysis manager receives all logs from the victim host and the Internet Emulator, shuts down and refreshes the victim host, and sends traffic logs to the dummy client generator.
(6) The dummy client generator inspects the received traffic logs to eliminate high risk communications, such as port scanning and remote exploitation, and generates the dummy client which emulates remaining communications. How to generate the dummy client is described in Section 3.5.2.
(7) The dummy client is executed in the environment with the real Internet connection. It repeats malware communication with the remote servers in the Internet and keeps collecting server responses and stores it in the data spool.

3.5 Implementation

We show the overview of the implementation of the proposed sandbox analysis system in **Fig. 2**. In the figure, the solid arrows indicate the communication by the analyzed malware and the dotted arrows indicates communication by the sandbox system for its operation.

3.5.1 Components

We implemented the entire system in a single real machine except the environment for the dummy client to be executed. We used a virtual machine by VMware Player 3.1.4 with Windows XP Professional SP1 running, as the victim host. The host OS is Ubuntu 11.04, on which the analysis manager, the access controller, the Internet Emulator, the dummy client generator are implemented. The network between the victim host and the host OS is realized by a virtual private network provided by VMware Player. Each component is implemented as follows:

- Victim Host

The victim host is implemented as a virtual machine running Windows XP SP1. To avoid VMware detection, we changed the default port number of VMware Server's Console and MAC address of the virtual NIC. Presently, basic monitoring tools such as Regmon [22] and Filemon [22] are installed in the victim host. We can also deploy other monitoring tools such as InCtrl [24] or techniques like API hooking [11]. Also, the victim host is configured to automatically download and execute a Windows batch file command.bat from the analysis manager upon each boot-up using SSH. The batch file contains the further instructions to be followed by the victim host:

(1) Downloading the malware sample
(2) Starting designated monitoring tools
(3) Executing the sample
(4) Sending the monitoring results to the manager after designated time period

As the command.bat file can be modified by the analysis manager, the procedure of the victim host can be easily controlled by the manager remotely.

- Internet Emulator

The Internet Emulator consists of two subcomponents: dummy servers and HitS. Both of them run on the host OS of the system. The dummy DNS, IRC, HTTP, HTTPS, NTP, SMTP, FTP, ECHO, and Feedback servers are implemented as light-weighted

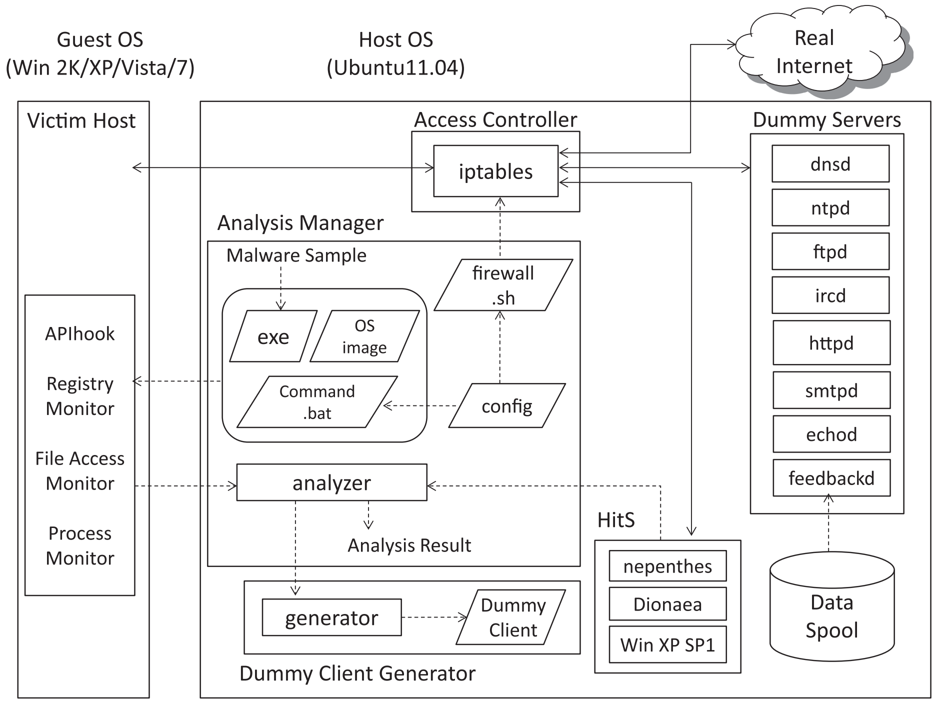

Fig. 2 Implementation of proposed system.

simplified server scripts by Perl to emulate the network services in the real Internet. When received A record query from a malware, the DNS server checks whether the responses from the server of the queried domain name has stored in the data spool or not. If the responses exist, the dummy DNS server returns IP address from the specific range. Otherwise, the dummy DNS server works as the DNS proxy. In addition, IP addresses which the dummy DNS server returns are unique during the same analysis pass. The Feedback server is the server to send the collected server responses to malware. When malware attempts to access the remote servers, the Feedback server obtains the domain name of the server by issuing a DNS query for the PTR record, and checks whether the server can be accessed. If the server can be accessed, the Feedback server works as the proxy server. Otherwise, the Feedback server searches the server responses correspond to the domain name, received data, and collected time in the data spool. The Feedback server has two response modes. One is the latest response mode and the other is the unique response mode. In the latest response mode, the Feedback server searches the latest collected server response and sends it to the malware. Also, the Feedback server can send the response collected at specific time. We can analyze the latest malware behaviors by using the latest response mode. Also we can analyze how the malware works at the specific time. In the unique response mode, in the case of HTTP protocol, the Feedback server searches and lists the unique responses and sends these responses to malware by rotation, and in the case of IRC protocol, the Feedback server searches the latest collected response and unique PRIVMSGs, and sends them to malware. We can analyze the malware various behaviors corresponding to the responses efficiently by using the unique response mode. In addition, in the case of IRC protocol, the Feedback server replace nickname appropriately.

HitS is dedicated to inspect the connections initiated by the sample to see if they contain any harmful attacks. In order to do it, HitS is designed to run emulated/real vulnerable network services for the sample to exploit, like a honeypot. Our current implementation uses a low-interaction honeypot program Nepenthes v0.2.2 [1] and Dionaea [21] as it can emulate multiple vulnerable services. We can also deploy a virtual/real machine running a full vulnerable OS to detect zero-day exploits although such an implementation is our future work.

- Access Controller

The access controller is implemented by iptables, a packet filtering application program. Before analysis, a shell script, called firewall.sh, is generated and executed by the analysis manager to apply newly generated filtering rules. All traffic from the victim host is redirected to either the real Internet or the Internet Emulator according to the filtering rules. For the connections to the Internet Emulator, we change their destination IP addresses to that of the host OS by REDIRECT target of PREROUTING chain in iptables so that they are sent to the servers running on the host.

- Analysis Manager

The analysis manager loads the configuration, and based on it, changes the configuration of the Internet Emulator and generates a Windows batch file command.bat and a shell script firewall.sh.

- Dummy Client Generator

The dummy client generator inspects the traffic logs and removes high risk communications, and generates the dummy client which replays the communications. There are several choices of how to replay these communications. For example, we can generate a dummy client for each malware sample to emulate its communications to remote servers. This way, we can collect the server responses corresponding to each malware. However, when the different malware accesses the same server with the same query, redundant data are saved in the data spool, and quantity of data increases. Therefore, for efficiency, we implemented a method to generate a single dummy client that emulates all unique communications observed by the execution of all malware samples.

The process of the dummy client generator is described in Section 3.5.2.

3.5.2 Generating Dummy Client

Generating a dummy client that emulates communications observed by malware sandbox analysis is done as follows:

(1) The dummy client generator divides the traffic logs into all TCP and UDP sessions. A session means a series of packets exchanged between a port of the victim host and a port of a remote host in the analysis.

(2) Each reconstructed session is closely inspected in the following rules to eliminate high risk communications. We call the rules *Attack Filtering Rules*.

- i. Do not allow any session whose source IP address is spoofed.
- ii. Do not allow any session that is determined to be a port scan by the inspection. Likewise, do not allow any session that is determined to be a part of a DoS attack by the inspection. In order to detect a DoS attack, it counts the number of sessions initiated by the victim host for the same destination IP address and port. If more than a threshold number Th_{DoS} of sessions are initiated, we consider it as a DoS attack. In order to detect a port scan, it counts the number of unique IP addresses accessed by the victim host on each destination port without DNS name resolution. We consider a port is scanned if more than a threshold number Th_{ps} of distinct IP addresses are accessed on that port by the victim host.
- iii. Do not allow any session that caused a successful exploitation of the vulnerabilities in HitS.
- iv. Do not allow any session whose application protocol is not authorized. The recognition of application protocol is based on the message flow analysis. Namely, we check if certain messages that characterize the application protocol, such as methods in HTTP and commands in IRC, are transmitted in the legitimate order of the protocol.

(3) The dummy client generator extracts the payload from each of all sessions which passed the above rules. The extracted payload is appended information of destination (e.g., domain name or destination IP address, destination port, protocol) and stored in the host OS.

(4) Each session is inspected under the following rules to increase efficiency. We call the rules *Duplication Reduction Rules*.

- i. Leave only one session if there are two or more sessions which send the same payload to the same server.
- ii. Delete a session whose payload is included in other session.
- iii. Leave only one session if there are two or more sessions which join the same IRC server with different IRC nick names. The reason of to use this filtering rule is that malware often communicates with IRC server by a random nickname, and if emulate each of all sessions, the number of emulate sessions will greatly increase.
- iv. Leave only one session if there are two or more sessions that send the same HTTP GET queries with different the arguments to the same server. The reason of to use this filtering rule is that malware sends the information about infected host and own id and etc. by using arguments of HTTP GET query, and if emulate each of all sessions, the number of emulate sessions will greatly increase.

(5) The dummy client generator generates the dummy client which replays the remaining communications. The dummy client emulates the communication with remote servers in the following procedures:

- i. The dummy client loads the destination information (i.e., domain name or destination IP address, destination port, protocol) of each session.
- ii. If the destination information includes a domain name, the dummy client resolves the domain name and obtains the corresponding destination IP address.
- iii. If the protocol is TCP, the dummy client tries to connect to the destination IP address on the destination port by 3-way handshake. When the connection is established, the dummy client sends the first packet to the destination. If the protocol is UDP, the dummy client simply sends the first packet to the destination IP address on the destination port.
- iv. The dummy client waits for a response from the destination. When receives the response, the dummy client stores it in the data spool.
- v. If the packet to be sent to the destination is still left and no response for the previous packet has arrived for a certain period of time, the dummy client sends the next packet to the destination.
- vi. After having sent all packets, the dummy client waits for the response from the server unless it is closed a session by the server and stores the received response in the data spool. In addition, in the case of the IRC protocol, the dummy client automatically replies PONG command when received PING command from the IRC server for that not to be closed session by the server.

4. Experiment

We conducted experiments to evaluate the proposed method using 7,184 malware samples captured in the wild by low-interaction honeypot (Nepenthes and Dionaea) and high-interaction honeypot and client honeypot and so on. The number of the names of the samples obtained from Symantec was 557. We divided the samples in two sample sets randomly. Then, we used one for creating dummy client and used the other for comparison. We call the former set training set and the later set test set. **Table 1** is a list of the top 20 virus names of samples in the each set.

The experiment has twofold purpose. First, we confirm whether we can observe the behaviors of remote servers by using the dummy client. Second, we confirm that we can improve the observability of the proposed sandbox than a simple Internet-

Table 1 Top 20 names of samples in each set obtained from symantec.

training set		test set	
Symantec	samples	Symantec	samples
W32.Virut.W	571	W32.Virut.W	597
W32.Spybot.Worm	347	W32.Spybot.Worm	321
W32.Virut.B	281	W32.Virut.B	271
Trojan Horse	189	W32.IRCBot	170
W32.Sality.AE	147	Trojan Horse	153
W32.Pinfi	139	W32.Sality.AE	143
W32.IRCBot	127	W32.Pinfi	130
W32.Korgo.S	125	W32.Rahack.W	127
W32.Rahack.W	125	W32.Korgo.S	120
W32.Virut.U	93	W32.Virut.U	104
W32.Gobot.A	89	W32.Korgo.V	90
W32.Korgo.V	85	Hacktool	76
Hacktool	80	W32.Gobot.A	72
Backdoor.Trojan	54	W32.Rahack.H	63
W32.Rahack.H	51	Backdoor.Trojan	62
Trojan.Gen	42	Trojan.Gen	48
Backdoor.Graybird	38	W32.Korgo.W	36
Suspicious.IRCBot	32	Suspicious.IRCBot	35
W32.IRCBot.Gen	31	W32.IRCBot.Gen	32

Table 2 Configuration of proposed sandbox.

EXECUTION_TIME	60 [sec]
VICTIM_HOST_OS	Windows XP Professional SP1
FILEMON	OFF
REGMON	OFF
HitS	Nepenthes v0.2.2
Dummy Servers	HTTP, NTP, IRC, FTP, SMTP, DNS, ECHO, Feedback
Threshholds	$Th_{ps} = 50$, $Th_{DoS} = 50$
Authorized Application	HTTP, IRC
Execution interval of the dummy client	1 [hour]

connected sandbox.

4.1 Procedure

In order to evaluate the proposed method, we prepared two sandboxes for comparison: the proposed sandbox with accumulated server responses collected by dummy clients and an Internet-connected sandbox. We call the former sandbox proposed sandbox and the latter Internet-connected sandbox. We show the configuration of the proposed sandbox in **Table 2**. Each sandbox is implemented on a single real machine with the same specification: Intel XEON 2.66 GHz $\times$ 4 with a main memory of 4 GB, and we implemented Internet-connected sandbox by removing Dummy Client Unit and Feedback server in the proposed sandbox.

The analysis by the proposed sandbox was done as follows:

(1) (First Analysis) First, we analyzed the training set in the Internet connected sandbox from Aug. 5 to Aug. 9, 2011.

(2) (Collection of Server Responses) Then, we generated a dummy client with the observed traffic logs, and executed the dummy client to collect the sever responses. The dummy client accessed each of the remote servers every hour from Sep. 6 to Oct. 25, 2011.

(3) (Second Analysis) Finally, we analyzed test set in the proposed sandbox and in the Internet-connected sandbox from Nov. 10 to Nov. 15, 2011.

4.2 Results

(1) First Analysis

Out of the 3,592 samples, 2,322 samples attempted to connect to remote hosts with 4,848,791 sessions consisting of 4,848,791 TCP sessions and 463 UDP sessions (except for DNS queries). Among them, 4,846,095 sessions (99.94%) were filtered out by Attack Filtering Rules. Most of the filtered sessions were considered as port scans. The remaining 2,696 sessions were again reduced to 242 sessions by Duplication Reduction Rules. Most of the session reduction was due to duplicated IRC sessions. Likewise, the number of destinations the samples attempted to connect was 4,617,688. Here, a destination means either a destination IP address or a domain name of the destined host. The number of destinations was reduced to 148 by Attack Filtering Rules. After all, 242 different types of sessions consisting of 224 HTTP sessions and 18 IRC sessions to 148 distinct destinations were selected to be replayed by the dummy client to obtain server responses.

(2) Collection of Server Responses

We ran the dummy client from Sep. 6 to Oct 25, 2011 to collect responses from the servers. The client accessed the 148 destinations by replaying the 242 different types of sessions every hour. Results are as follows:

A) For 70 types of replayed sessions, the client always received the same response with the same content every hour.

B) For 23 types of replayed sessions, the client received no response during the experiment.

C) For 37 types of replayed sessions, the client received either a response with the same content or no response.

D) For 112 types of replayed sessions, responses from the server changed during the experiment.

In Case (C), connectibility of the servers changes depending on when the client accesses the servers. In Case (D), responses from the servers changes depending on when the client accesses the servers. Thus, the result of normal sandbox analysis of malware which attempt to access the remote servers with these sessions may be influenced by analysis time. Here, the change of the response means that the hash value of received HTTP content is changed or the IRC PRIVMSG is changed.

Windows Executable Files

Figure 3 shows the number of new Windows executable files collected by the dummy client over time. Here, we count only new executables based on their hash values. From the figure, we confirmed that the dummy client could continuously receive a number of new Windows executable files. As downloaded executables can be used for further malicious activities, it is important to observe such a behavior.

IRC PRIVMSG

Figure 4 shows the number of new IRC PRIVMSG collected by the dummy client over time. From the figures, we confirmed that the dummy client could continuously receive a number of new PRIVMSGs from the IRC-based C&C servers. As PRIVMSG is often used as a C&C command between a bot and its herder, it is important to observe such commands. For example, there are messages such as "!get http://netnetnet1.com/sd7.txt", which seems to be a command for downloading a file. Actually, in the sandbox analysis, after having received such a message, the malware accessed the URL and downloaded an executable file.

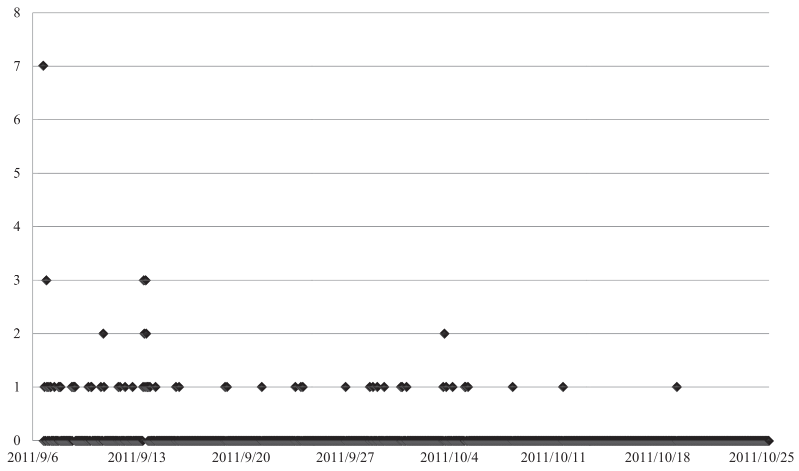

Fig. 3 Number of newly observed windows executable files.

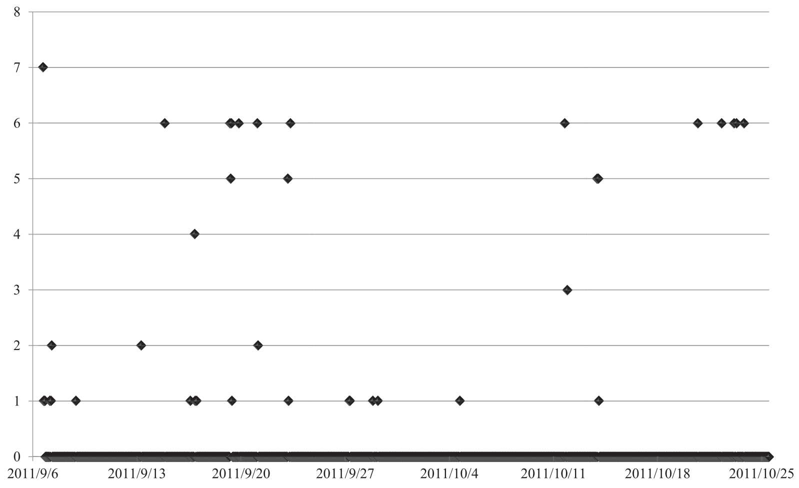

Fig. 4 Number of newly observed PRIVMSG.

Table 3 Comparison of proposed sandbox and internet-connected sandbox.

	Internet-Connected Sandbox	Proposed Sandbox	unit
Samples that used TCP	1982	2073	(sample)
Samples that used HTTP	1310	1354	(sample)
Samples that used HTTP GET	1308	1344	(sample)
Samples that used HTTP POST	84	203	(sample)
Samples that received EXE by GET	310	438	(sample)
Samples that used SMTP	78	162	(sample)
Samples that used IRC	448	480	(sample)
Samples that used IRC PASS	29	30	(sample)
Samples that used IRC NICK	445	472	(sample)
Samples that used IRC USER	445	472	(sample)
Samples that used IRC JOIN	439	440	(sample)
Samples that used IRC MODE	43	46	(sample)
Samples that received IRC PRIVMSG	404	409	(sample)
# of unique received PRIVMSGs	8	37	(PRIVMSG)
Samples that used UDP	2089	2067	(sample)
Samples that used DNS	2088	2063	(sample)
Samples that used DNS A-record queries	2088	2063	(sample)
Samples that used DNS MX-record queries	18	142	(sample)
Samples that used DNS PTR-record queries	30	128	(sample)
# of unique queried domain names	687	1683	(domain)
Queried domain names per sample	2.2	4.3	(domain)
Samples that performed port scan	1619	1720	(sample)
Samples that performed DoS attack	40	86	(sample)
Samples that sent shell code to 135/TCP	26	23	(sample)
Samples that sent shell code to 139/TCP	175	273	(sample)
Samples that sent shell code to 445/TCP	1503	1591	(sample)
Samples that sent shell code to 3127/TCP	79	79	(sample)
Size of traffic log per sample	11.1	75.7	(MB)

(3) Second Analysis

Table 3 shows the comparison between the analysis results of the proposed sandbox and the Internet-connected sandbox. In Table 3, almost all items increased by the proposed sandbox. Especially, numbers of samples that used HTTP POST, samples that received executable file by HTTP GET, samples that used SMTP, samples that used DNS MX-record queries and PTR-record queries, unique queried domain names, and samples that performed port scan greatly increased. These results indicate an improved observability of malware behaviors by the proposed sandbox. In the proposed sandbox, 1,347 malware samples attempted to connect the 19 remote servers whose responses have been collected by the dummy client. And, out of the 19 remote servers, 8 servers could not be accessed in the second analysis, thus the Feedback server replied to it using the collected responses. We consider that this is one of the main reasons for this improvement.

The above results demonstrate that the proposed sandbox has certain level of feasibility to observe malware behavior corresponding to the behavior of remote servers.

5. Discussion

Although the proposed sandbox showed a possibility to work for observing malware behavior according to server behavior, there are several limitations that need to be discussed:

5.1 Detection of Dummy Client by Remote Servers

In this implementation, the dummy client simply replays the same communication observed in the sandbox. Such a simple replay can be easily defeated by the attacker. The attacker can include sensitive information (source IP address, timestamp, other internal system parameter, or their hash values) into URL parameters, and reject queries with bad parameters. Moreover, if the remote servers change responses according to URL parameters, we can't get the all responses because we eliminate some sessions by applying Duplication Reduction Rules. In addition, the attacker can adopt SSL or any other kind of protocols that are highly interactive. In order to deal with these interactive protocols, we also need to make our dummy client more interactive. In order to obtain necessary information for more complete emulation of server-client interactions, we will need to look into the internal behavior of the malware samples by techniques like API hooking.

In addition, in the experiment, the dummy client used the same IP address for accessing the remote servers. Therefore, the attacker can detect the dummy client by looking at the frequently used IP addresses of the clients connecting their servers. Therefore, to avoid this network-based detection of dummy client, we should frequently change the IP address of the client. Another solution is to use anonymity networks like The Onion Router (Tor) [10]. Tor relays a multi-layered encrypted message among its Onion Routers for sender/receiver anonymity. Namely, by using TOR, a dummy client can connect to the remote servers without revealing its IP address. However, we need to consider carefully from the viewpoint of attackers. Although TOR reasonably provides sender anonymity, there are several techniques for the receiver to determine if the sender is using TOR [8], [26]. If a normal malware victim hardly uses TOR, the very usage of TOR can raise an attacker's suspicion. A similar discussion applies in the case of an anonymous proxy [4].

5.2 Determining Changes of Server Behavior

In the proposed method, it is important to determine whether remote servers have changed its behavior in order to emulate remote servers effectively. However, it is not a trivial issue. There are indeed some parts of the server responses that changes every time a client accesses it such as a time stamp. We need to automatically distinguish these changes from those caused by the attacker's behavior in order to efficiently perform the feedback analysis. Moreover, the attacker can disturb our decision by intentionally randomizing responses of his server or introducing a challenge-response protocol to avoid a replayed query. However, we should remark that for stable connection attackers sometimes use a paid server hosting service that supports only a standard protocol and in that case, such customized protocols cannot be used. In fact, we confirmed that some of the remote servers we observed were hosted by a paid service and they used only a regular HTTP protocol and thus our method worked effectively.

5.3 Filtering High-Risk Communications

In our proposed method, the dummy client keeps replaying the observed malware traffic. It means that the client might mistakenly keep replaying a high-risk communications, such as remote exploits and DoS, to innocent hosts. In addition, although we don't filter HTTP traffic in the experiments, there are various exploits by using HTTP query such as SQL injection, RFI, and comment spamming. Therefore, the filtering of high-risk communication is a critical issue in our proposal. Besides the presented filtering rules, we can also estimate the likeliness of the remote server to be an attacker's server in various ways. One possible solution is to use a blacklist of malicious servers. More detailed discussion on the attack filtering is our future work.

6. Conclusion

We proposed a novel sandbox analysis method using a dummy client, an automatically generated script to interact with the remote servers instead of the actual sample. In the proposed method, at first we execute a malware sample in the sandbox. Secondly, we inspect the malware traffic observed in the sandbox and filter out high-risk communications. The rest of the traffic data is then used by the dummy client to interact with the remote servers. Then the dummy client continually interacts with the remote servers instead of the sample itself and effectively collects the responses from the servers. The collected server responses are then fed back to the Internet Emulator in the sandbox and will be used for improving observability of malware sandbox analysis. The experiment using in-the-wild samples showed a possibility that the server response collections and feedback analysis can reveal more behavior of malware.

References

[1] Baecher, P., Koetter, M., Holz, T., Dornseif, M. and Freiling, F.C.: The Nepenthes Platform: An Efficient Approach to Collect Malware, *Proc. 9th International Conference on Recent Advances in Intrusion Detection* (*RAID 2006*), pp.165–184 (2006).

[2] Bailey, M., Oberheide, J., Andersen, J., Mao, Z.M., Jahanian, F. and Nazario, J.: Automated Classification and Analysis of Internet Malware, *Proc. 10th International Conference on Recent Advances in Intrusion Detection* (*RAID 2007*), pp.178–197 (2007).

[3] Bayer, U., Kruegel, C. and Kirda, E.: TTAnalyze: A Tool for Analyzing Malware, *Proc. 15th Annual Conference of the European Institute for Computer Antivirus Research* (*EICAR*) (2006).

[4] Brozycki, J.: Detecting Anonymous Proxy Usage (2008), available from ⟨http://www.sans.edu/student-files/presentations/detecting_anonymous_proxies_handouts.pdf⟩.

[5] Canallero, J., Grier, C., Kreibich, C. and Paxson, V.: Measuring Pay-per-Install: The Commoditization of Malware Distribution, *Proc. 20th USENIX Security Symposium* (2011).

[6] Caballero, J., Johnson, N.M., McCamant, S. and Song, D.: Binary code extraction and interface identification for security applications, *Proc. 17th Annual Network and Distributed System Security Symposium* (*NDSS 2010*) (2010).

[7] Caballero, J., Poosankam, P., Kreibich, C. and Song, D.: Dispatcher: Enabling active botnet infiltration using automatic protocol reverse-engineering, *Proc. 16th ACM Conference on Computer and Communications Security* (*CCS 2009*), pp.621–634 (2009).

[8] Chakravarty, S., Stavrou, A. and Keromytis, A.D.: Identifying Proxy Nodes in a Tor Anonymization Circuit, *Proc. 2008 IEEE International Conference on Signal Image Technology and Internet Based Systems*, pp.633–639 (2008).

[9] Cho, C.Y., Babic, D., Shin, E.C.R. and Song, D.: Inference and Analysis of Formal Models of Botnet Command and Control Protocols, *Proc. 17th ACM Conference on Computer and Communication Security* (*CCS2010*), pp.426–439 (2010).

[10] Dingledine, R., Mathewson, N. and Syverson, P.: Tor: The Second-Generation Onion Router, *Proc. 13th USENIX Security Symposium*, pp.303–320 (2004).

[11] Father, H.: Hooking Windows API – Technics of Hooking API Functions on Windows, *CodeBreakers Journal*, Vol.1, No.2 (2004).

[12] Inoue, D., Yoshioka, K., Eto, M., Hoshizawa, Y. and Nakao, K.: Malware Behavior Analysis in Isolated Miniature Network for Revealing Malware's Network Activity, *IEEE International Conference on Communications* (*ICC 2008*), pp.1715–1721 (2008).

[13] Kolbitsch, C., Holz, T., Kruegel, C. and Kirda, E.: Inspector GAdget: Automated Extraction of Proprietary Gadgets from Malware Binaries, *Proc. IEEE Symposium on Security and Privacy* (May 2010).

[14] Moser, A., Kruegel, C. and Kirda, E.: Exploring Multiple Execution Paths for Malware Analysis, *IEEE Symposium on Security and Privacy*, pp.231–245 (2007).

[15] Miwa, S., Miyachi, T., Eto, M., Yoshizumi, M. and Shinoda, Y.: Design and Implementation of an Isolated Sandbox with Mimetic Internet Used to Analyze Malwares, *Proc. DETER Community Workshop on Cyber Security Experimentation and Test* (2007).

[16] Yoshioka, K., Kasama, T. and Matsumoto, T.: Sandbox Analysis with Controlled Internet Connection for Observing Temporal Changes of Malware Behavior, *The 4th Joint Workshop on Information Security* (*JWIS 2009*) (2009).

[17] Yoshioka, K. and Matsumoto, T.: Multi-Pass Malware Sandbox Analysis with Controlled Internet Connection, *IEICE Trans. Fundamentals*, Vol.E93-A, No.1, pp.210–218 (2010).

[18] Willems, C., Holz, T. and Freiling, F.: Toward Automated Dynamic Malware Analysis Using CWSandbox, Security & Privacy Magazine, Vol.5, No.2, pp.32–39, IEEE (2007).

[19] Anubis, available from ⟨http://analysis.seclab.tuwien.ac.at/⟩.

[20] CWSandbox, available from ⟨http://www.cwsandbox.org/⟩.

[21] dionaea – catches bugs, available from ⟨http://dionaea.carnivore.it/⟩.

[22] FileMon and RegMon for Windows, available from ⟨http://technet.microsoft.com/en-us/sysinternals/⟩.

[23] Joebox, available from ⟨http://www.joebox.org/⟩.

[24] InCTRL Reporting & Analysis, Measuretronix Ltd., available from ⟨http://www.measuretronix.com/en/products/inctrl-reporting-analysis⟩.

[25] NORMAN Sandbox Information Center, available from ⟨http://www.norman.com/microsites/nsic/⟩.

[26] Tor or not Tor: How to tell if someone is coming from a Tor exit node, in PHP, available from ⟨http://www.irongeek.com/i.php?page=security/detect-tor-exit-node-in-php⟩.

Takahiro Kasama received his B.E. and M.E. degrees in Computer Engineering from Yokohama National University in 2009 and 2011, respectively. He is currently a doctor course student at the Graduate School of Environment and Information Sciences, Yokohama National University. He is currently a researcher at the National Institute of Information and Communications Technology, Japan. His research interest covers a wide area of network security including malware analysis. He received the Best Paper Award at the Computer Security Symposium 2010 (CSS2010), and the IPSJ Yamashita SIG Research Award in 2011.

Katsunari Yoshioka received his B.E., M.E. and Ph.D. degrees in Computer Engineering from Yokohama National University in 2000, 2002, 2005, respectively. From 2005 to 2007, he was a Researcher at the National Institute of Information and Communications Technology, Japan. Currently, he is an Associate Professor for Division of Social Environment and Informatics, Graduate School of Environment and Information Sciences, Yokohama National University. His research interest covers a wide range of information security, including malware analysis, network monitoring, intrusion detection, and information hiding. He was awarded 2009 Prizes for Science and Technology by The Commendation for Science and Technology by the Minister of Education, Culture, Sports, Science and Technology.

Tsutomu Matsumoto is a Professor of Division of Social Environment and Informatics, Graduate School of Environment and Information Sciences, Yokohama National University. His current roles include an Associate Member of the Science Council of Japan and a core member of CRYPTREC — the Cryptography Research and Evaluation Committees for governmental use of cryptographic technology. Starting from design and analysis of various cryptosystems and cryptographic protocols in the early 80's, he has opened up the field of security measuring for logical and physical security mechanisms including human-machine cryptography, information hiding, software security, biometric security, side-channel security, and artifact-metrics. He got Doctor of Engineering degree from the University of Tokyo in 1986. He received the Achievement Award from IEICE in 1995, the DoCoMo Mobile Science Award in 2006, the Culture of Security Award in 2008, and the Prize for Science and Technology, the Commendation by the Minister of Education, Culture, Sports, Science and Technology in 2010.

Masaya Yamagata received his M.E. degree from the Graduate School of Science and Technology, Shinshu University, Japan, in 1998. He joined NEC Corporation in 1998, and has been engaged in research on network security mechanisms such as firewalls, intrusion detection, and role-based flow management.

Masashi Eto received his LL.B degree from Keio University in 1999, received the M.E. and Ph.D. degrees from Nara Institute of Science and Technology (NIST) in 2003, 2005, respectively. From 1999 to 2003, he was a system engineer at Nihon Unisys, Ltd., Japan. He is currently a researcher at National Institute of Information and Communications Technology (NICT), Japan. His research interests include network monitoring, intrusion detection, malware analysis, and auto-configuration of the Internetworking. He received the Best Paper Award at the 2007 Symposium on Cryptography and Information Security (SCIS 2007).

Daisuke Inoue received his B.E. and M.E. degrees in electrical and computer engineering and Ph.D. degree in engineering from Yokohama National University in 1998, 2000 and 2003, respectively. He joined the Communications Research Laboratory (CRL), Japan, in 2003. The CRL was relaunched as the National Institute of Information and Communications Technology (NICT) in 2004, where he is the director of Cybersecurity Laboratory in Network Security Research Institute. His research interests include security and privacy technologies in wired and wireless networks, incident analysis and response technologies based on network monitoring and malware analysis. He received the best paper award at the 2002 Symposium on Cryptography and Information Security (SCIS 2002), the best paper award at the 2nd and 3rd Joint Workshop on Information Security (JWIS 2007 and 2008), and the commendation for science and technology by the minister of MEXT, Japan, in 2009.

Koji Nakao received his B.E. degree of Mathematics from Waseda University, in Japan, in 1979. Since joining KDDI in 1979, he has been engaged in the research on communication protocol, and information security technology for telecommunications in KDDI laboratory. After 2003, he has moved to KDDI head office to construct and manage its security systems. In 2004, he has started to additionally work for NICT (National Information Communication Technologies). His present positions are "Information Security Fellow" to manage all the security issues required in KDDI and "Distinguished Researcher" to manage research activities for network security technologies in NICT. He received the IPSJ Research Award in 1992, METI Ministry Award and KPMG Security Award in 2006, and Contribution Award (Japan ITU), NICT Research Award, Best Paper Award (JWIS) and MIC Bureau Award in 2007 and The Commendation for Science and Technology by the Minister of Education, Culture, Sports, Science and Technology (Prizes for Science and Technology: Research Category) in 2009. He is a member of IPJS and IEICE. He has also been a part-time instructor in Waseda University and Nagoya University.