IPSJ Journal Vol. 52 No. 3 1144–1159 (Mar. 2011)

Regular Paper

Your Sandbox is Blinded: Impact of Decoy Injection to Public Malware Analysis Systems

Katsunari Yoshioka,[†1] Yoshihiko Hosobuchi,[†1]
Tatsunori Orii[†1] and Tsutomu Matsumoto[†1]

The use of public Malware Sandbox Analysis Systems (public MSASs) which receive online submissions of possibly malicious files or URLs from an arbitrary user, analyze their behavior by executing or visiting them by a testing environment (i.e., a sandbox), and send analysis reports back to the user, has increased in popularity. Consequently, anti-analysis techniques have also evolved from known technologies like anti-virtualization and anti-debugging to the detection of specific sandboxes by checking their unique characteristics such as a product ID of their OS and a usage of certain Dynamic Link Library (DLL) used in a particular sandbox. In this paper, we point out yet another important characteristic of the sandboxes, namely, their IP addresses. In public MSASs, the sandbox is often connected to the Internet in order to properly observe malware behavior as modern malware communicate with remote hosts in the Internet for various reasons, such as receiving command and control (C&C) messages and files for updates. We explain and demonstrate that the IP address of an Internet-connected sandbox can be easily disclosed by an attacker who submits a decoy sample dedicated to this purpose. The disclosed address can then be shared among attackers, blacklisted, and used against the analysis system, for example, to conceal potential malicious behavior of malware. We call the method Network-based Sandbox Detection by Decoy Injection (NSDI). We conducted case studies with 15 representative existing public MSASs, which were selected from 33 online malware analysis systems with careful screening processes, and confirmed that a hidden behavior of the malware samples was successfully concealed from all of the 15 analysis systems by NSDI. In addition, we found out the risk that a background analysis activity behind these systems can also be revealed by NSDI if the samples are shared among the systems without careful considerations. Moreover, about three months after our first case study it was reported that a real-world NSDI was conducted against several public MSASs.

†1 Yokohama National University

1. Introduction

As malware, such as computer viruses, worms, bots, Trojan horses and spyware, have become serious and critical threats that affect our lives significantly, great efforts have been made to analyze their behavior in detail to develop effective countermeasures.

Malware sandbox analysis is considered a promising approach for the analysis of malware which involves the execution of an unknown program in a testing environment, known as a sandbox, to observe and analyze its behavior. With the growing popularity of malware sandbox analysis, there are a number of systems [1)–9)] that use a public interface to accept online submissions of samples from arbitrary users, automatically analyze them using a sandbox, and send analysis reports back to the user. Similar systems also exist for the analysis of suspicious web sites [1),10)–20)]. In this paper, we refer to such a malware analysis system with a public interface as a *public Malware Sandbox Analysis System* (*public MSAS*). It was reported that a public MSAS received over 900,000 submissions of unique samples (based on MD5 hashes) in less than two years [21)], demonstrating the popularity of public MSASs.

The sandbox for public MSASs can be either *isolated or Internet-connected*. As the name suggests, the isolated sandbox is not connected to the Internet and instead utilizes emulated network services thus can analyze samples safely [7),22)–25)]. However, the drawback is that it cannot fully observe the communications of samples with remote hosts in the Internet such as command and control (C&C) and malware download servers, which can provide useful information for their analysis. In contrast, the Internet-connected sandbox [21),26)–30)] can observe such communications although there is a risk that attacks from the executed sample may exit the sandbox and cause harm to other hosts in the Internet.

As malware analysis technologies become known, malware authors have begun to utilize anti-analysis techniques, such as anti-virtualization [31)–35)] and anti-debugging [31),32)], to detect and disturb the analysis. Moreover, the recent popularity of the public MSASs has made the anti-analysis techniques evolve to detecting particular sandboxes by checking their unique characteristics such as the product ID of their OS and a usage of a certain Dynamic Link Library

(DLL) [36)–38)].

In this paper, we point out yet another important characteristic of the sandboxes, namely, their IP addresses. We explain that the IP address of an Internet-connected sandbox can be easily disclosed by an attacker who submits a decoy sample dedicated to this purpose. The disclosed address can then be shared among attackers, blacklisted, and used against the analysis system, for example, to conceal the potential malicious behavior of malware. In this paper we call the method *Network-based Sandbox Detection by Decoy Injection (NSDI)*. We conducted case studies with 15 representative existing public MSASs, which were selected from 33 online malware analysis systems with careful screening processes, and confirmed that a hidden behavior of the malware samples was successfully concealed from all of the 15 analysis systems by NSDI. In addition, we found out the risk that even background analysis activities of these systems can be revealed by NSDI if the samples are shared among them without careful considerations.

Moreover, on October 22nd, 2009, Vitaly Kamluk from Kaspersky Lab reported that a real-world NSDI attack had been conducted against several public MSASs [39)]. The issue date of the article was about 1.5 months after the submission of our first technical report [40)] indicating the possibility of NSDI and four days before its official publication. This paper is a comprehensive version of the report [40)] covering definitions on the model of public MSAS, its properties, definition of NSDI, and additional case studies.

The rest of the paper is organized as follows: Section 2 explains the related works. Section 3 describes the model of public MSASs and their properties. Section 4 explains sandbox detection techniques including NSDI. Section 5 describes the case study with the nine existing public MSASs for evaluating the impact of NSDI. Section 6 discusses the mitigation of NSDI. Finally, Section 7 gives the conclusion.

2. Related Works

In recent years, malware sandbox analysis has been studied intensively and can be categorized into two approaches in terms of Internet connectivity: one involves a completely isolated sandbox while the other uses a sandbox with an Internet connection. An example of the former approach is the Norman Sandbox [7)], which emulates a network environment and does not allow any connection to the Internet. The Norman Sandbox emulates many network services such as HTTP, FTP, SMTP, DNS, IRC, and P2P. Other works [22)–25)] take a similar approach to provide emulated network services. The limitation of this approach is that it is difficult to emulate the actual Internet, as malware utilize various forms of communications. When they connect to a server, such as a C&C or file server, for an update under the attacker's control, they could use arbitrary or even customized protocols [41),42)] for data transmission and authentication which makes the service emulation increasingly challenging. The second approach is to carefully connect the sandbox to the actual Internet. Examples of Internet-connected sandboxes include the CWSandbox [5),8),29)] and Anubis [1),21),27)] and other sandbox analysis systems used in Refs. 26), 28), 30), 43).

With the growing popularity of malware sandbox analysis, a number of public MSASs exist [1)–20)]. Although most systems accept a Windows executable, there are several systems that support the analysis of Javascript [19)], Flash files [38)], Dynamic Link Library (DLL) files [6)], and Portable Document Format (PDF) files [6),19)]. In addition, public MSASs exist that analyze web sites [1),10)–20)]. Presently, these services are provided free of charge. A recent paper [21)] provides interesting statistics from the analyses of over 900,000 unique samples (based on MD5 hashes) which were submitted to their public MSAS in less than two years.

As malware analysis technologies became known, malware authors have begun to utilize anti-analysis techniques, such as anti-virtualization [31)–35)] and anti-debugging [31),32)], to detect and disturb the analysis. These techniques focus on, for instance, the difference in the way certain instructions are handled with the presence of virtual machines [34)], unique configurations in OS and applications with a specific virtualization solution such as VMware [32)], or the difference in CPU instruction execution time [32),33),44)]. Moreover, the recent popularity of the public MSASs has made the anti-analysis techniques evolve to detecting a specific sandbox by checking their unique characteristics such as a product ID of their OS [21),36)–38)] and a usage of a certain Dynamic Link Library (DLL) [36)–38)]. We note that all above techniques are host-based detection methods in the sense that the detection is indeed carried out in the targeted host.

There are also a number of network-based anti-analysis techniques. One of the

techniques that has been deployed by real malware in the wild is the Internet connectivity check. That is, some malware try to connect to certain well-known sites, such as google.com, to see if it is connected to the Internet so that they can detect an isolated analysis environment [24]. Also, a downloader, which contains only a piece of code that downloads its real payload from a remote server, also can blind an isolated sandbox. In Ref. 31), a network-based virtual machine detection method is proposed that focuses on the values of TCP time stamp of the packets generated by a targeted remote host. In Ref. 45), a network-based detection method is proposed in which malware launches a test attack to see if it is in an analysis environment with the assumption that the analysis environment will not allow the attack out.

3. Public Malware Sandbox Analysis Systems

3.1 Models

Public MSASs can be classified into two groups; one is for analyzing a submitted file and the other is for analyzing a web site of a submitted URL. In this paper, we call the former a *public MSAS for Sample Files* (*public MSAS-F*, for short) and the latter a *public MSAS for Web Sites* (*public MSAS-W*, for short).

At first, we explain the model of public MSAS-F. **Figures 1** and **2** depict the models of a public MSAS-F with an isolated sandbox and that with an Internet-connected sandbox, respectively. In both systems, the *submitter* is the user of the analysis system who submits a sample file (e.g., Windows executable) to the system. The *reception* is the publicly known interface of the system, which is typically realized by a web site to accept sample submissions. The *sandbox* represents the testing environment where the submitted sample is executed and analyzed. As explained in the previous section, an isolated sandbox does not connect to the real Internet and instead connects to the emulated Internet, which consists of various dummy servers as depicted in Fig. 1. On the other hand, the Internet-connected sandbox connects to the Internet as in Fig. 2. The outbound traffic from an Internet-connected sandbox is checked carefully in order to mitigate the risks of infections outside the sandbox [28),30)]. The sandbox implements various means, such as API hooking, to monitor the internal behavior of the executed sample. Finally, the *analysis report* that describes the detailed malware

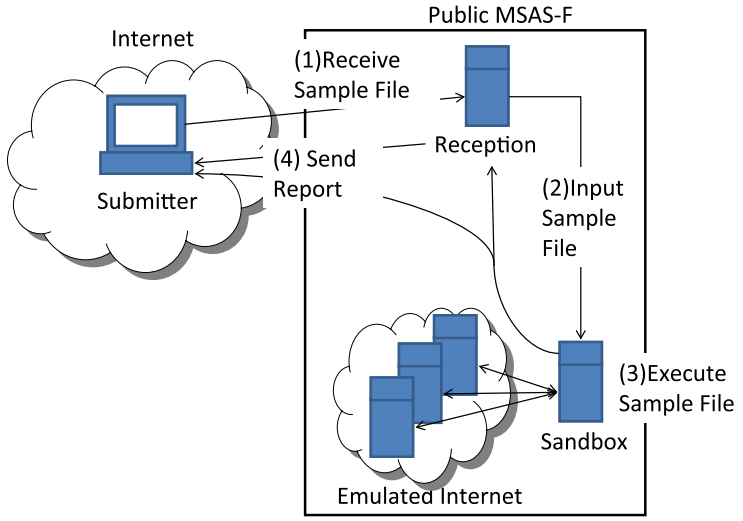

Fig. 1 A model of public malware sandbox analysis system for sample files (public MSAS-F) with an isolated sandbox.

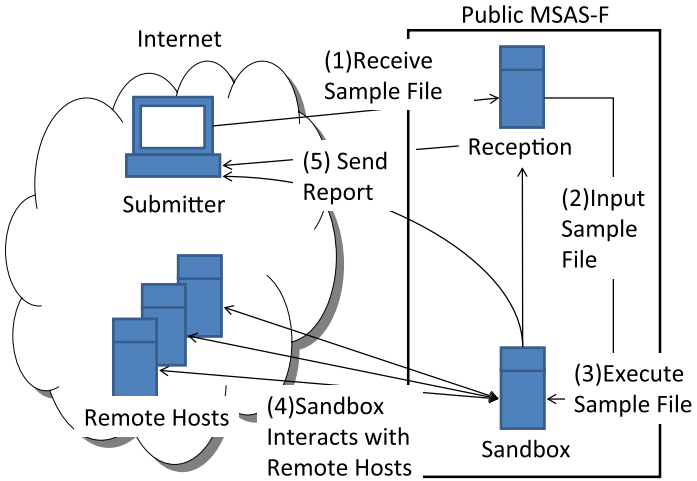

Fig. 2 A model of public MSAS for sample files (public MSAS-F) with an internet-connected Sandbox.

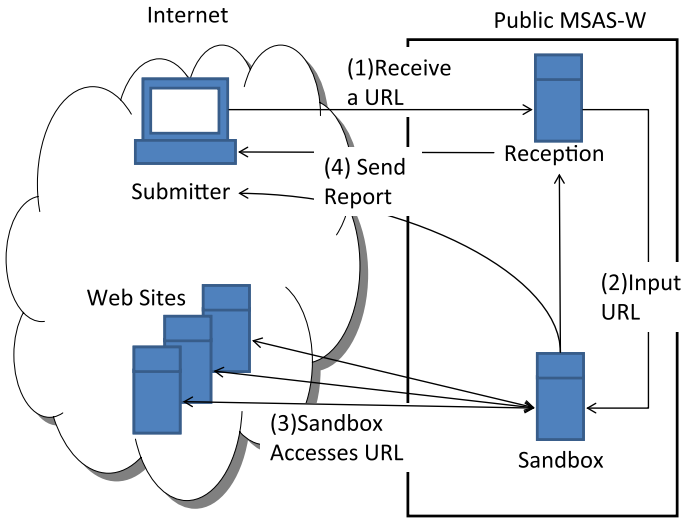

Fig. 3 A model of public MSAS for web sites (public MSAS-W).

behavior such as API calls, file access, registry access, process creation, network activities, is provided to the submitter via reception or other electronic means, such as an email.

Second, we explain the model of a public MSAS-W. As depicted in **Fig. 3**, the analysis of a web site works similarly as a public MSAS-F. The submitter first submits a to-be-tested URL to the system. Then, the sandbox actually accesses the web site to obtain the web contents for analysis. In order to analyze a web site which refers to other web sites, the sandbox also connects to them to obtain the referred content. Therefore, the sandbox for a public MSAS-W is Internet-connected by nature.

3.2 Properties

We consider three important properties of public MSASs, namely, observability, containment, and efficiency.

Observability is the property for providing sufficient information regarding the behavior of the submitted file or web site. In general, the purpose of malware sandbox analysis is to learn how tested malware would behave in real production systems so that effective countermeasures can be developed to protect the production systems. The isolated sandboxes suffer from the fact that they are not connected to the Internet while real production systems are likely to be connected to it. Also, the sandbox detection techniques described in this paper significantly affect the observability of a public MSAS as some malware would not behave the same as in production systems if they detect that they are being analyzed in a sandbox. We note that observability of a public MSAS for its provider and that for its submitter are usually not equal as the submitter only obtains an analysis report of the submitted sample, which is normally a digest of all the monitoring results obtained by the analysis system during the execution of the sample.

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 is the property for constantly providing analysis results with sufficient information in a reasonable amount of time.

There are trade-offs between these properties. For example, one extreme policy to achieve the highest level of containment of outgoing attacks is to completely isolate the sandbox from the outside networks, however, this strategy will greatly reduce observability. Likewise, providing no analysis report to the submitter will achieve the highest level of containment of system information although it will provide no observability to the submitter. Another example of a trade-off is that executing the same malware sample multiple times can increase the observability as it may reveal malware's probabilistic behavior although the efficiency is decreased.

4. Sandbox Detection by Decoy Injection

In this section, we explain several sandbox detection methods that can be conducted against public MSASs. The detection methods consist of two phases. In the first phase, termed *Decoy Injection* (DI), the attacker submits a decoy to

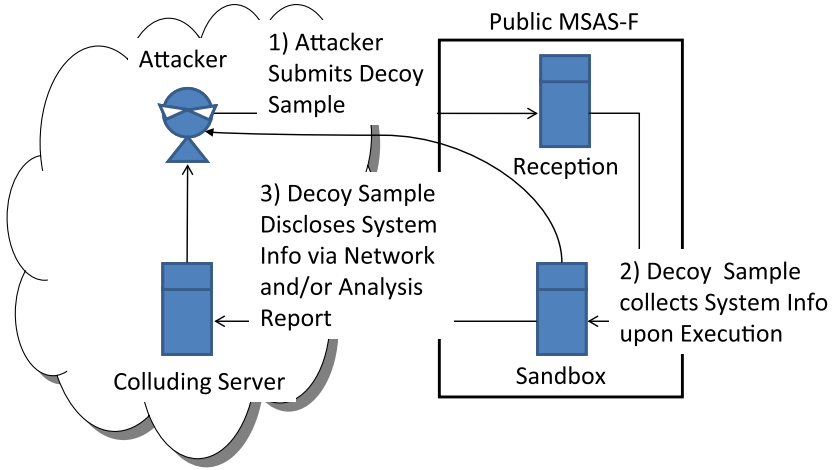

Fig. 4 Decoy injection against public MSAS-F.

the targeted public MSAS in order to disclose its system information. Then, in the second phase, the disclosed information is utilized by the attacker to detect the sandbox. In Section 4.1, we explain the first phase, namely, DI. In Section 4.2, we explain host-based sandbox detection methods that have been observed in the wild. Then, in Section 4.3, we explain a novel network-based sandbox detection method, namely NSDI.

4.1 Decoy Injection

Since public MSASs are available to arbitrary users, even an attacker can use them as a submitter to learn the systems. The basic idea of DI is to submit a *decoy sample* that collects and discloses information of the targeted public MSAS. Typically, unique characteristics of the sandbox such as a product ID of its OS or the existence of certain files, registry keys, and processes can be utilized for sandbox detection.

Figure 4 depicts the overview of DI against a public MSAS-F. In order to disclose the system information from a public MSAS-F, the attacker simply submits a decoy sample to the targeted system. The decoy sample is eventually executed in the system and discloses the system information it has collected. It should be noted that there are two possible channels over which the system information can be disclosed. The first channel is communication between an Internet-connected sandbox and remote hosts. Namely, the decoy sample can simply send the collected information over the network to a designated remote host under the attacker's control, called a *colluding server*, outside the sandbox. Obviously, this channel does not exist in a public MSAS using an isolated sandbox. The second channel is the analysis report. Namely, the attacker submits a decoy sample that "embeds" the obtained system information into the analysis report. For example, a decoy sample can encode a Windows' product ID and use it as the name of a file it creates. Since the name of the files created by the sample is likely to be mentioned in the analysis report, the attacker can obtain the product ID via the report. This channel exists even in a public MSAS using an isolated sandbox.

Figure 5 depicts an overview of DI against a public MSAS-W. First, the attacker prepares a web server, called a *colluding web server*. The colluding web server contains an attack script that exploits a vulnerability of the web browser used by the sandbox in order to get a decoy sample downloaded and executed in the sandbox. Then, the attacker submits a *decoy URL*, which is indeed the URL of the colluding web server, to the target public MSAS-W. The sandbox of the system eventually accesses the colluding web server to obtain the web contents. Since the web contents contain the exploit, the sample is downloaded and executed in the sandbox. The executed decoy sample then works the same as in the case for a public MSAS-F.

4.2 Host-based Sandbox Detection

As the second phase, the attacker utilizes the disclosed system information to detect a sandbox. Recently, it has been reported that some malware started enhancing a mechanism to detect a sandbox of specific public MSASs [36)–38)]. One of the detection methods is to get a snapshot of the loaded modules to see if it contains a certain module used only in a specific public MSAS. Another method is to get a product ID of the Windows OS by checking the registry key "HKLM¥Software¥Microsoft¥Windows¥CurrentVersion." Since these sandbox detection methods are carried out by the sample in the sandbox itself, we call them host-based sandbox detection.

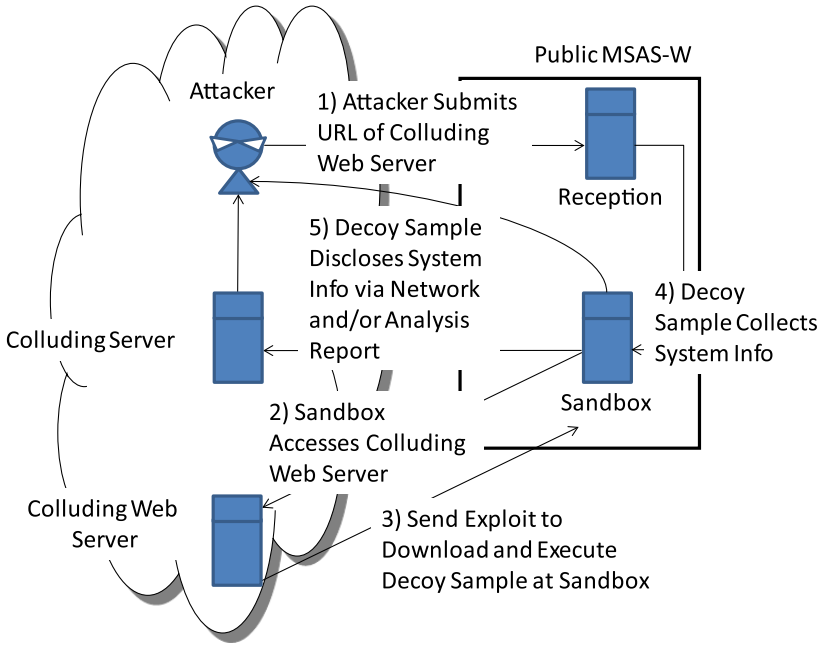

Fig. 5 Decoy injection against public MSAS-W.

4.3 Network-based Sandbox Detection by Decoy Injection

In this subsection, we explain NSDI. That is, we show that the IP address of an Internet-connected sandbox can also be leveraged by an attacker to detect the sandbox with cooperation between the decoy sample and the colluding server.

Figure 6 depicts the first phase of NSDI for obtaining the IP address of the sandbox. We first explain NSDI against a public MSAS-F. At the beginning, the attacker creates a decoy sample that connects to the colluding server upon its execution. When connecting to the colluding server, the sample uses a hidden *identifier* in order for the attacker to distinguish its connection from other traffic at the colluding server side. The identifier can be in any form as long as it is recognizable by the attacker at the colluding server. For example, the identifier can be hidden in a typical malware communication, such as a file request or connection to an IRC channel (e.g., a file name in HTTP GET request, nick

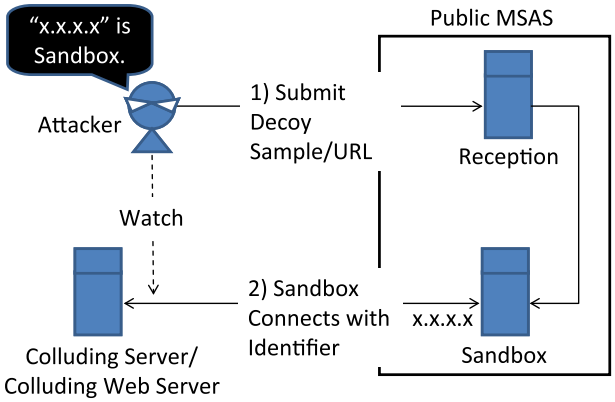

Fig. 6 First Phase of Network-based Sandbox Detection by Decoy Injection.

name or channel name of IRC). After preparing the decoy sample, the attacker submits it to the targeted public MSAS. Eventually, the sample is executed in the sandbox and connects to the colluding server with the hidden identifier. As the submitted sample is the only possibility that would connect to the server using the identifier, the attacker can learn the IP address of the sandbox.

In the second phase of NSDI, the attacker leverages the disclosed IP address. A typical scenario, depicted in **Fig. 7**, is that the attacker who maintains a botnet with a C&C server uses the disclosed IP address to distinguish a bot being analyzed in the sandbox from those really infected a production system.

Next, we explain NSDI against a public MSAS-W. The attacker first generates *a decoy URL*. The decoy URL consists of the domain name of the colluding web server and a name of the requested file as an identifier. For example, if the domain name of the colluding web server is "col.com" and the identifier is "abc001", then the decoy URL becomes "http://col.com/abc001.html." After generating a decoy URL, the attacker submits it to the target public MSAS-W. The sandbox in the system then accesses the colluding web server and requests the file abc001.html. The attacker knows that the sandbox is the only possibility of connecting the server using the identifier and therefore considers the IP address used for the connection as that of the sandbox. Note that NSDI for a public MSAS-W is simpler for an attacker to conduct than the regular DI explained in Fig. 5 since

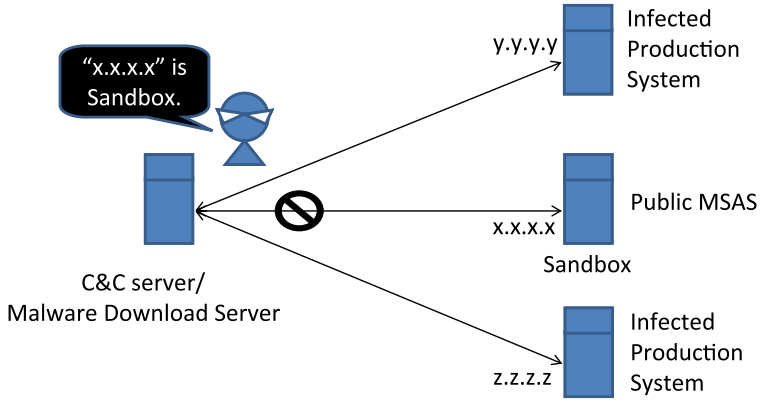

Fig. 7 Second Phase of Network-based Sandbox Detection by Decoy Injection.

the only system information necessary for the detection is the IP address of the sandbox. Therefore, the attacker does not have to prepare any attack script on his colluding web server nor have the decoy sample downloaded and executed at the sandbox, unlike the case explained in Fig. 5.

After learning the IP address of the sandbox, the attacker can distinguish the sandbox trying to examine his malicious web site from real victims. For example, he could selectively send benign web contents back to the sandbox while providing malicious contents to other regular clients.

5. Case Studies

In this section, we describe our case studies to evaluate the impact of NSDI on 15 existing public MSAS, consisting of eight public MSAS-F and seven public MSAS-W. An important observation regarding NSDI is that the longer the targeted public MSAS uses the same IP address for its sandbox, the more effective the detection becomes. The summary of our findings is as follows: (1) NSDI was successful against all 15 public MSASs, 11 of which appeared to have utilized an Internet-connected sandbox with nearly unchanged IP addresses and the other four utilized an isolated sandbox, and (2) Background activities of the MSASs, such as automated surveillance of the remote servers and forwarding of samples from one system to another, were made visible by NSDI.

5.1 Preparation

5.1.1 Selecting Target public MSASs

In order to decide the target public MSASs for our case studies, we first searched for online malware analysis systems in the WWW as well as a series of academic literatures to find the candidates. We tried to cover as many sectors as possible from academic institutes and universities, security vendors, non-profit research groups to individual researchers. Also, we tried to cover as many countries/regions as possible for the location of the providers of the target systems. We found 33 online malware analysis systems as a result of our search. Since it relates to the vulnerability of real systems in operation, we will not disclose their actual names in this paper. Then, we performed a screening with the following three conditions to decide the target systems for our case studies:

1. A target system should describe itself as a sandbox analysis system (or at least its analysis report implies a utilization of sandbox/dynamic analysis method that involves actual execution of samples).
2. A target system should provide a detailed analysis report that is specific enough for us to check the effectiveness of NSDI.
3. A target system should be operated stably so that an analysis report can be obtained from each decoy submission for fair evaluation.

We have set the first condition since there are a number of online malware analysis systems that do not actually execute the submitted sample but instead statically scan it (e.g., using signature-based pattern matching and other heuristic-based detection). Also, there are many website-check systems that do not actually access the to-be-tested website but instead use blacklist and accumulated database for deciding its maliciousness (i.e., web reputation). These systems are clearly not public MSAS by definition and not in our scope.

We have set the second condition because some online analysis systems do not provide a detailed analysis report but only their final decision as to whether the submitted file/URL is malicious or not. However, in order to strictly confirm the effectiveness of NSDI, we needed to look at the detailed analysis report to check whether the hidden behavior (or the hidden web page $\mathrm{HTML_M}$) was successfully concealed by NSDI. Therefore, we included only systems that provide a detailed analysis report as the targets of our experiments.

Finally, we set the third condition because some online analysis systems were not operated stably at the time of our case studies. We were occasionally not able to submit a sample to these unstable systems because of system failure (e.g., reception web server was down, etc.) or obtained no analysis reports from them. In order to avoid inconsistency in the experimental results, we omitted such unstable systems from our evaluations.

As the results of the screening of the 33 existing online malware analysis systems, 15 systems, consisting of eight public MSAS-F and seven public MSAS-W, were determined to satisfy all three conditions and selected as the target systems. The target systems include most well-known public MSASs and are also rich in diversity, provided by academic institutes and universities, security vendors, or even individual security researchers in 9 different countries. Please see the appendix for the concrete result of our screening of the 33 systems.

5.1.2 Decoy Samples and Colluding Server for Evaluation of public MSAS-F

For evaluating the eight public MSAS-F, we first prepared a decoy sample that behaved as follows:

1. It attempts to connect to the colluding server. If the domain name of the server is not resolved or there is no reply from the colluding server, the sample simply halts.
2. When connected to the colluding server, it issues an HTTP GET request. The name of the requested file represents the unique identifier of the sample. It halts if a TCP session is not established after a certain period of time.
3. If it receives the requested file from the server, the sample checks whether the file contains a keyword ASCII string.
4. If a keyword string is found in the file, the sample proceeds with its hidden behavior. Namely, it creates a file on the infected system and tries to connect to another server. If a keyword string is not found, the sample halts.

There are two main features of the decoy sample:

- Connecting to the colluding server with a unique identifier to disclose the IP address of the sandbox, and
- Showing its hidden behavior by the trigger from the colluding server (i.e., receiving a file with a keyword string).

Next, we prepared a colluding server which maintains two types of lists. The first is a blacklist of the IP addresses used by the sandboxes of the targeted analysis systems, while the second is a list of valid identifiers, which includes the names of files that the decoy samples would request when executed. The server works as follows:

1. It waits for an incoming TCP connection request from a client.
2. If an HTTP GET request arrives from a client, the server checks whether the name of the requested file is in the list of valid identifiers.
3. If the identifier is valid, the server includes the IP address of the client into a blacklist. It also sends a file with the keyword string back to the client if configured to do so.

There are two main features of the colluding server:

- Verifying an incoming HTTP GET request to confirm they are from the decoy samples and updating the blacklist, and
- Triggering hidden behavior of the decoy samples by sending a file with a keyword string. The trigger can be switched on or off as desired.

Figure 8 depicts how the decoy sample and colluding server work together. After being submitted to the target public MSAS-F, the decoy sample is executed in the sandbox and sends an HTTP GET request with its associated identifier to the colluding server. The colluding server then examines whether each incoming HTTP GET request is from the submitted decoy sample or not by checking the identifier. If the identifier is valid, it considers that the request is indeed from the submitted sample and includes the IP address of this client to the blacklist. After the address is blacklisted, the server has a choice of whether or not to send a file with a keyword string back to the decoy sample. This file works as a trigger for the decoy sample to proceed with its hidden behavior. We intend to test if malware behavior can be controlled at the attackers' will by checking whether the hidden behavior is reported by the targeted MSAS in its analysis report.

5.1.3 Decoy URLs and Colluding Web Server for Evaluation of public MSAS-W

We prepared a colluding web server and a set of decoy URLs for the evaluation of the seven public MSAS-W. The decoy URLs were generated as described

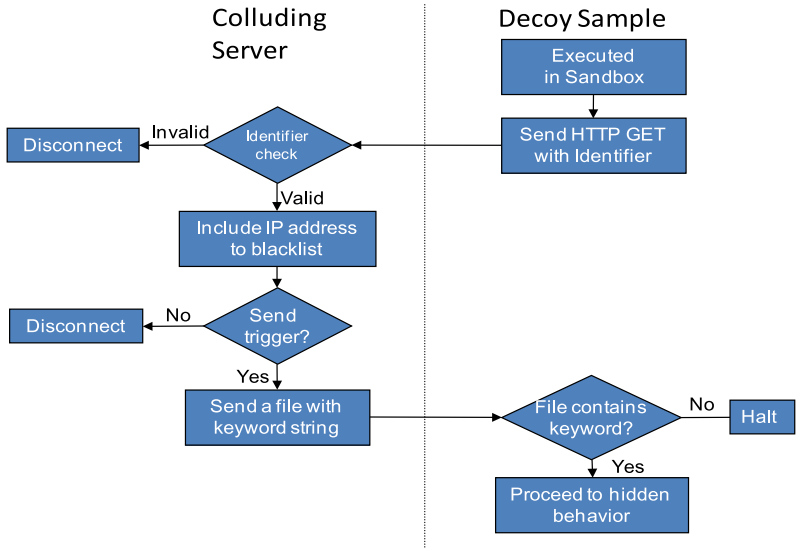

Fig. 8 Flow chart of the process by decoy sample and colluding server for case study.

in Section 4.3. The colluding web server maintains two types of lists as in the evaluation of public MSAS-F. The first is a blacklist of the IP addresses used by the sandboxes of the targeted analysis systems, while the second is a list of valid identifiers, which includes the names of files that are used in the decoy URLs. The server contains two HTML files: HTML_M and HTML_B that redirect to two different sites: Site_M and Site_B. We assumed that Site_M is malicious and Site_B benign and then test if the attacker can indeed hide the "malicious" redirection by selectively sending HTML_B back to the targeted public MSAS-W. Note that it was good enough for us to check which redirected site was visited by the sandbox for this evaluation and therefore we did not have to make Site_M really malicious. The server works as follows:

1. It waits for an incoming TCP connection request from a client.
2. If an HTTP GET request arrives from a client, the server checks whether the name of the requested file is in the list of valid identifiers.
3. If the identifier is valid, the server includes the IP address of the client into the blacklist and sends either one of the two HTML files back to the client.

There are two main features of the colluding web server:

- Verifying an incoming HTTP GET request to confirm they are from the sandbox and updating the blacklist, and
- Selectively sending back either one of the HTML files to confirm that the "malicious" redirection can be hidden from the targeted public MSAS-W.

5.2 Procedures

The actual case studies were carried out in two different time periods.

The first case study was performed during a one-week period in July, 2009 with nine target public MSASs. We refer to the i-th system as System i for $i = 1, 2, .., 9$. Note that Systems 1 to 7 are public MSAS-F and Systems 8 and 9 are public MSAS-W. Five decoy samples/URLs per day per system were submitted to the nine target systems. All submitted samples/URLs contained a distinct identifier. We prepared a colluding server to which all the submitted decoy samples could be connected. We configured the colluding server to send a trigger to the sample to proceed with its hidden behavior only if the IP address appeared for the first time in the experimental period. Likewise, we prepared a colluding web server for all decoy URLs. Then, we configured the colluding web server to send HTML_M only if the IP address appeared for the first time in the experimental period.

In September, 2010, in order to increase the coverage of our case study, we performed the second case study with six other public MSASs in the same manner as the first case study. Note that these six systems were completely different from the previously evaluated nine systems, Systems 1 to 9. We refer to them as System 10, System 11, to System 15. Note that System 10 is a public MSAS-F and Systems 11, 12, 13, 14, 15 are public MSAS-W. The second case study was done during a one-week period in September, 2010.

5.3 Results

Summary The results of the case studies are summarized in **Table 1**. In short, NSDI was successful against all evaluated public MSASs. Note that we consider that NSDI against a public MSAS-F was successful only if the hidden behavior of all submitted decoy samples was indeed successfully hidden from the system. In other words, no hidden behavior was reported in the analysis report except when the trigger was intentionally sent. Likewise, we considered that NSDI

Table 1 Summary of Experiments.

Target System	Experimental Period	Type of public MSAS	# of Submitted Decoy Samples or URLs	# of Observed Identifiers at Colluding Server	# of Observed IP Addresses	Speculated Sandbox Type	NSDI
1	July, 2009	public MSAS-F	35	35	1	Internet-connected	Successful
2	July, 2009	public MSAS-F	35	0	0	Isolated	Successful
3	July, 2009	public MSAS-F	35	35	2	Internet-connected	Successful
4	July, 2009	public MSAS-F	35	0	0	Isolated	Successful
5	July, 2009	public MSAS-F	35	35	10 in a /24 NW	Internet-connected	Successful
6	July, 2009	public MSAS-F	35	0	0	Isolated	Successful
7	July, 2009	public MSAS-F	35	35	1	Internet-connected	Successful
8	July, 2009	public MSAS-W	35	35	12 in a /24 NW	Internet-connected	Successful
9	July, 2009	public MSAS-W	35	35	1	Internet-connected	Successful
10	Sep, 2010	public MSAS-F	35	0	0	Isolated	Successful
11	Sep, 2010	public MSAS-W	35	35	1	Internet-connected	Successful
12	Sep, 2010	public MSAS-W	35	35	14 in two /29 NWs	Internet-connected	Successful
13	Sep, 2010	public MSAS-W	35	35	1	Internet-connected	Successful
14	Sep, 2010	public MSAS-W	35	35	4 in a /29 NW	Internet-connected	Successful
15	Sep, 2010	public MSAS-W	35	35	1	Internet-connected	Successful

against a public MSAS-W was successful only if all hidden redirection to Site_M was indeed successfully hidden from the system. In other words, the redirection to Site_M was never reported in the analysis report except when we intentionally reveal it.

Systems with Fixed IP Address We observed HTTP GET requests at our colluding server immediately after each decoy submission to 11 target systems, Systems 1, 3, 5, 7, 8, 9, 11, 12, 13, 14, and 15. Thus, we consider that they utilize an Internet-connected sandbox. Among them, 6 systems, namely Systems 1, 7, 9, 11, 13, and 15, used a single IP address respectively for the entire experimental period of one week. It implies that these systems are extremely vulnerable to NSDI since an attacker could have detected their sandboxes for one week by submitting just a single decoy. System 3 is slightly better than the other systems as it actually changed the IP address of its sandbox during the experiment. However, an attacker could still have easily followed the change of the IP address by submitting a few decoys per week. Consequently, the analysis reports obtained from these systems during the experiment never revealed the hidden behavior of the submitted decoy.

Systems with Fixed IP Address Range Systems 5, 8, 12, and 14 used more IP addresses than the others. However, these addresses belonged to certain fixed subnets such as /24 and /29. We can reasonably assume that the systems had assigned these IP address ranges for their sandboxes. In fact, some of the subnets were assigned to the organizations of the system owners and anyone could have confirmed that with public database such as whois. An attacker would have needed to submit a few decoys to figure out that the target systems utilize such an address block. However, once the address block is known to the attacker, it can be blacklisted and the attacker can detect the sandbox with very low risk of falsely blacklisting the real infected machines.

Systems utilizing Isolated Sandbox There were no connection attempts to our colluding servers using identifiers for the four target systems, Systems 2, 4, 6, and 10. We consider these systems utilize an isolated sandbox. As expected, these systems never revealed the hidden behavior of the submitted decoys since the decoys never received a trigger from our colluding servers. These systems are incompetent against NSDI and other network-based sandbox detection [24),25)].

We now explain our additional findings. We observed a number of interesting

Table 2 Revealed Background Activities of Evaluated Public MSASs.

Case	Identifier	# of IP Addresses	Description of Observed Activities	Our Speculation
1	System 3	1	Received HTTP GET Request 1.5 Hrs. After Every Decoy Sample Submission	Automated Background Surveillance by Analysis System
2	System 2	1	Received HTTP GET Request After Every Decoy Sample Submission	Automated Background Surveillance by Analysis System
3	System 2	1	Received HTTP GET Request using WGET Every 1.5 Hrs. for 3 Days after Every Decoy Sample Submission	Automated Background Surveillance by Analysis System
4	System 5	3	Received a few HTTP GET requests	Manual/Auto Surveillance by Analysis System
5	System 5	1	HTTP GET Request using IP Address of System 7	Samples Passed from One System to Another
6	System 5	2	HTTP GET Request using IP Addresses used in Cases 2 and 3	Samples Passed from One System to Another

background activities of the evaluated systems, which are summarized in **Table 2**.

Automated Background Surveillance from System 3 Approximately 1.5 hours after each submission of a decoy sample to System 3, we observed an HTTP GET request using the same identifier (Case 1). The format of the HTTP GET requests differed from that of the decoy samples. Thus, we consider that these requests did not originate from the executed samples but were generated by a type of automated background surveillance tool deployed by the system. The IP address used by this possible surveillance was fixed, which implies a high risk of detection by the attacker.

Automated Background Surveillance from System 2 We observed two patterns of automated accesses using identifiers for System 2. Concerning the first pattern, following each submission of a decoy sample to System 2, we observed an HTTP GET request using the same identifier (Case 2). The format of the requests differed from that of the decoy samples. Therefore, we consider that these requests were generated by an automated surveillance tool. As only a single IP address was used for this surveillance, it could have also been detected by an attacker. In the second pattern of automated access, we observed continuous HTTP GET requests after every submission of a decoy sample (Case 3) to System 2. Nearly every 1.5 hours, we received an HTTP GET request that appeared to be issued by WGET [46], a tool for file acquisition. These requests continued for three days after each submission. Again, only one IP address was used for this possible surveillance.

Possible Manual/Automated Surveillance from System 5 We observed several, distinct HTTP GET requests using identifiers for System 5 (Case 4) which may represent manual/automated surveillance by the system. Three different IP addresses were used for these access attempts.

Sample Passed from One System to Another In Case 5, we observed an HTTP GET request using an identifier for System 5. However, the IP address used for this access was the address of System 7 which indicates that the sample submitted to System 5 was somehow input into System 7. Likewise, in Case 6, we observed HTTP GET requests using an identifier for System 5 but with the IP addresses used in Cases 2 and 3. This indicates that samples submitted to System 5 were somehow submitted to System 2 and then eventually to the surveillance system of System 2.

Probing from Other Attackers In addition to the cases described above, we observed a few attempts to connect to colluding servers without the use of an identifier. However, it is unclear whether these access attempts were initiated from the analysis systems or unrelated attacker, such as an in-the-wild malware trying to locate a server to infect.

An overview of the background activities of the seven public MSAS-F is summarized in **Fig. 9**. From the figure, it can be seen that System 5 is working as a dispatcher of samples to other public MSAS-F. Also, several systems were deploying background surveillance tools in order to continuously monitor the remote servers. In particular, System 2 itself utilizes an isolated sandbox that does not connect to the Internet, while it utilizes two background surveillance tools that monitor the colluding server.

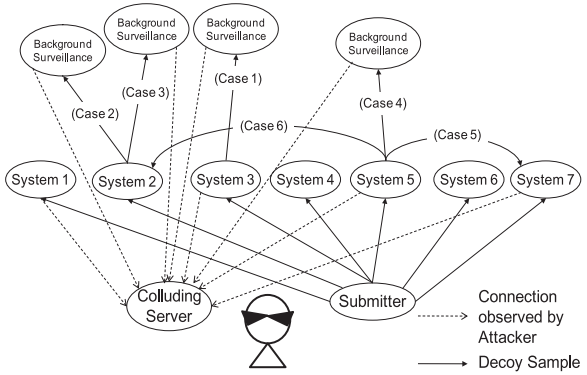

Fig. 9 Revealed background activities of evaluated public MSAS-F.

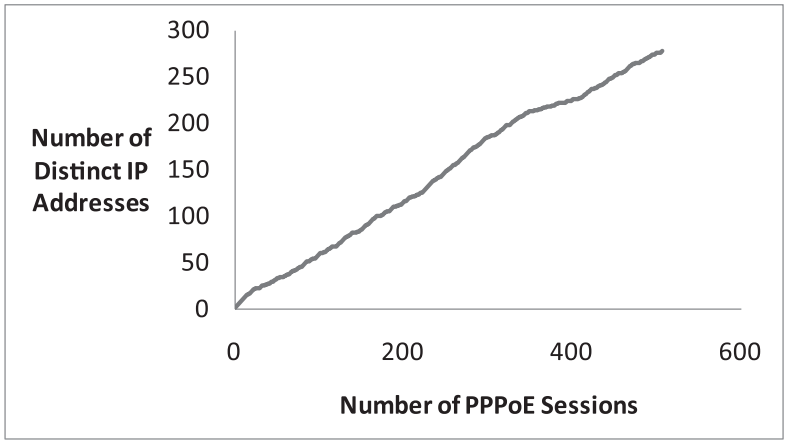

Fig. 10 Number of distinct IP addresses assigned to single subscriber of FTTH line from major Japanese ISP versus number of PPPoE sessions.

6. Discussion

In our case study, it was revealed that NSDI can be effectively conducted against existing public MSASs. In this section, we discuss several issues in favor of the mitigation of NSDI.

6.1 Using Time-Shared IP Addresses with Production Systems

It is obvious that the longer a public MSAS uses the same IP address for its sandbox, the more effective NSDI becomes. Therefore, the IP address of the sandbox should be frequently changed. However, since the operation cost of NSDI is so low, it is assumable that an attacker conducts it frequently enough to disclose and blacklist all used IP addresses. For preventing such a straightforward blacklisting by the attacker, we can deploy a time-sharing of IP addresses with real production systems. If an attacker mistakenly blacklists an IP address of a production system that is infected by malware, he will lose an opportunity to control the production system. As depicted in Fig. 7, the attacker's interest is effectively blocking only the sandbox while properly controlling the real infected systems. Therefore, time-sharing of the IP addresses with production systems will decrease the effectiveness of NSDI. The more production systems we time-share the addresses with, the more ineffective NSDI becomes.

A possible realization of the above idea is leveraging a dynamic IP address assignment by a commercial Internet Service Provider (ISP). Since many commercial ISPs hold a large number of IP addresses to be shared among the service subscribers, we can leverage this property for public MSASs. **Figure 10** shows the number of distinct IP addresses assigned to a single subscriber of a Fiber-to-the-Home (FTTH) line provided by a major Japanese ISP versus the number of PPPoE (Point-to-Point Protocol over Ethernet) sessions initiated by the subscriber. The figure shows that the ISP assigned a new IP address for approximately every two PPPoE sessions.

However, there are drawbacks in the usage of the ISP-provided addresses. First of all, ISPs may also filter specific traffic and we can not control how the filtering is done and therefore the analysis results may not be as reliable. Moreover, ISPs might ban the analysis service. However, since many victims of malware indeed use a commercial ISP, it is worth considering setting up an analysis system with the same network environment as the victims.

6.2 Using Anonymity Networks and/or Proxies

Another possibility for mitigating NSDI is the usage of anonymity networks like The Onion Router (TOR) [47]. TOR relays a multi-layered encrypted message among its Onion Routers for sender/receiver anonymity. Sender anonymity

allows a client to hide its identity from the server. Namely, by using TOR, a sandbox can connect to the colluding server without revealing its IP address. While this seems to be a perfect solution for NSDI, 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 [48),49)]. 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 [50)].

6.3 Service Provider's Awareness

Our case study demonstrated that even background analysis activities behind public MSASs can be revealed by NSDI. Also, we have seen several decoy samples passed from one system to another. This implies the risk in information sharing among the malware analysis community. A public MSAS has been a useful tool for the community to collect malware samples efficiently. However, we should all be aware that there is always the possibility that the submitted sample is a decoy and can disclose important information of our analysis system and our cooperator's system with whom we share the samples. This applies to any other online services that accept malware samples for analysis purpose such as virustotal [51)] and offensivecomputing [52)].

6.4 User's Awareness

Users of public MSASs should also be aware of anti-analysis techniques including NSDI and recall the fact that the behavior in the analysis report is what has been observed in a particular sandbox and does not always represent how the sample would behave in a real production system. Being aware of the fact and its limitation, public MSASs can still be a useful tool that instantly provides various information about the submitted samples.

7. Conclusion

We considered the possibility of a novel anti-analysis technique, called NSDI, which may be conducted against public MSASs. We conducted case studies with 15 representative existing public MSASs, which were selected from 33 online malware analysis systems with careful screening processes, and confirmed that a hidden behavior of the malware samples was successfully concealed from all of the 15 analysis systems by NSDI. The case study also showed the risk that even a background analysis activity can be revealed by the method. Our future works include the discussion on an effective mitigation of NSDI.

References

1) Anubis. http://analysis.seclab.tuwien.ac.at/
2) Autovin. http://autovin.pandasecurity.com.my/?page_id=332
3) BitBlaze. https://aerie.cs.berkeley.edu/
4) Comodo Instant Malware Analysis. http://camas.comodo.com/cgi-bin/submit
5) CWSandbox. http://www.cwsandbox.org/
6) Joebox. http://www.joebox.org/
7) Norman Sandbox. http://www.norman.com/technology/norman_sandbox/
8) Sunbelt CWSandbox: Malware Research Labs. http://www.sunbeltsecurity.com/
9) Threat Experts. http://www.threatexpert.com/
10) Aguse. http://www.aguse.jp/
11) Dr. Web online check. http://online.us.drweb.com/?url=1
12) gred. https://www.gred.jp/?tab=goleo
13) Jsunpack. http://jsunpack.jeek.org/dec/go
14) LinkScanner. http://linkscanner.explabs.com/linkscanner/default.aspx
15) Online Link Scan. http://onlinelinkscan.com
16) Unmask Parasite. http://www.unmaskparasites.com/
17) viCHECK.ca. https://vicheck.ca/
18) Webutation. http://www.webutation.org/
19) Wepawet. http://wepawet.iseclab.org/
20) vURL Online. http://vurldissect.co.uk/
21) Bayer, U., Habibi, I., Balzarotti, D., Kirda, E. and Kruegel, C.: A View on Current Malware Behaviors, *Proc. 2nd Usenix Workshop on Large-Scale Exploits and Emergent Threats, LEET'09* (2009).
22) 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).
23) Inoue, D., Yoshioka, K., Eto, M., Hoshizawa, Y. and Nalao, K.: Automated Malware Analysis System and its Sandbox for Revealing Malware's Internal and External Activities, *IEICE Trans.*, Vol.E92D, No.5, pp.945–954 (2009).
24) 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*, p.6 (2007).
25) Yoshioka, K., Inoue, D., Eto, M., Hoshizawa, Y., Nogawa, H. and Nakao, K.:

Malware Sandbox Analysis for Secure Observation of Vulnerability Exploitation, *IEICE Trans.*, Vol.E92D, No.5, pp.955–966 (2009).
26) Bailey, M., Oberheide, J., Andersen, J., Mao, Z.M., Jahanian, F. and Nazario, J.: Automated Classification and Analysis of Internet Malware, *Proc. Recent Advances in Intrusion Detection*, *RAID'07, LNCS*, Vol.4637, pp.178–197 (2007).
27) Bayer, U., Comparetti, P.M., Hlauschek, C., Kruegel, C. and Kirda, E.: Scalable, Behavior-Based Malware Clustering, *Symposium on Network and Distributed System Security* (*NDSS*) (2009).
28) Bayer, U., Kruegel, C. and Kirda, E.: TTAnalyze: A Tool for Analyzing Malware, *15th Annual Conference of the European Institute for Computer Antivirus Research* (*EICAR*) (2006).
29) Willems, C., Holz, T. and Freiling, F.: Toward Automated Dynamic Malware Analysis Using CWSandbox, *Security & Privacy Magazine*, IEEE, Vol.5, No.2, pp.32–39 (2007).
30) Yoshioka, K. and Matsumoto, T.: Multi-pass Malware Sandbox Analysis with Controlled Internet Connection, *IEICE Trans.*, Vol.E93A, No.1, pp.210–218 (2010).
31) Chen, X., Andersen, J., Mao, Z.M., Bailey, M. and Nazario, J.: Towards an Understanding of Anti-virtualization and Anti-debugging Behavior in Modern Malware, *The 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (*DSN 2008*) (2008).
32) Holz, T. and Raynal, F.: Detecting Honeypots and other Suspicious Environments, *Proc. 2005 IEEE Workshop on Information Assurance and Secrity* (2005).
33) Raffetseder, T., Kruegel, C. and Kirda, E.: Detecting System Emulators, *Proc. 10th Information Security Conference* (*SC*), *LNCS*, Vol.4779 (2007).
34) Rutkowska, J.: Red Pill... or how to detect VMM using (almost) one CPU instruction. http://invisiblethings.org/papers/redpill.html
35) Win32.agobot. http:/www3.ca.com/securityadvisor/virusinfo/virus.aspx?id=37776
36) Evil Fingers: Sandbox Awareness. http://evilfingers.blogspot.com/2009/01/sandbox-awareness.html
37) Evilcodecave's Weblog: OffensiveCOding updated — Emulation/AV Awareness. http://evilcodecave.wordpress.com/tag/cwsandbox/
38) OpenSC.ws: Detect 5 different sandboxes. http://www.opensc.ws/snippets/3558-detect-5-different-sandboxes.html
39) Kamluk, V.: A black hat loses control. http://www.securelist.com/en/weblog?weblogid=208187881
40) Yoshioka, K., Hosobuchi, Y., Orii, T. and Matsumoto, T.: Vulnerability of Malware Sandbox Analysis as an Online Service, *IPSJ* (*Information Processing Society of Japan*) *Computer Security Symposium 2009, Session F5-1* (Oct. 2009).
41) Porras, P., Saidi, H. and Yegneswaran, V.: A Foray into Conficker's Logic and Rendezvous Points, *Proc. USENIX Workshop on Large-Scale Exploits and Emergent Threats* (2009).
42) Rajab, M.A., Zarfoss, J., Monrose, F. and Terzis, A.: A Multifaceted Approach to Understanding the Botnet Phenomenon, *Proc. 6th ACM SIGCOMM Conference on Internet Measurement*, pp.41–52 (2006).
43) Sandboxie. http://www.sandboxie.com
44) Kirda, E., Kruegel, C., Banks, G., Vigna, G. and Kemmerer, R.: Behavior-based Spyware Detection, *Proc. 15th Conference on USENIX Security Symposium*, Vol.15, No.19 (2006).
45) Wang, P., Wu, L., Cunningham, R. and Zou, C.C.: Honeypot Detection in Advanced Botnet Attacks, *International Journal of Information and Computer Security 2010*, Vol.4, No.1, pp.30–51 (2010).
46) GNU WGET. http://www.gnu.org/software/wget/
47) Dingledine, R., Mathewson, N. and Syverson, P.: Tor: The Second-Generation Onion Router, *Proc. 13th USENIX Security Symposium* (2004).
48) Chakravarty, 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).
49) Tor or not Tor: How to tell if someone is coming from a Tor exit node, in PHP. http://www.irongeek.com/i.php?page=security/detect-tor-exit-node-in-php
50) Brozycki, J.: Detecting Anonymous Proxy Usage (2008). http://www.sans.edu/resources/student_presentations/detecting_anonymous_proxies_handouts.pdf
51) Virustotal. http://www.virustotal.com/
52) Offensive Computing: Community Malicious code research and analysis. http://www.offensivecomputing.net

Appendix

Table 3 Screening result of 33 candidate online malware analysis systems. Systems 1 to 15 were selected as target systems for our case studies and Systems 16 to 33 were not selected because of the reason described in each column.

System	Analysis Target	Condition 1 (Implication of Sandbox Analysis)	Condition 2 (Detailed Analysis Report)	Condition 3 (Stable Operation)	Our Final Decision If **Selected or Not** as Target
1	File	Yes	Yes	Yes	**Selected** as Target in Case Study 1
2	File	Yes	Yes	Yes	**Selected** as Target in Case Study 1
3	File	Yes	Yes	Yes	**Selected** as Target in Case Study 1
4	File	Yes	Yes	Yes	**Selected** as Target in Case Study 1
5	File	Yes	Yes	Yes	**Selected** as Target in Case Study 1
6	File	Yes	Yes	Yes	**Selected** as Target in Case Study 1
7	File	Yes	Yes	Yes	**Selected** as Target in Case Study 1
8	Website	Yes	Yes	Yes	**Selected** as Target in Case Study 1
9	Website	Yes	Yes	Yes	**Selected** as Target in Case Study 1
10	File	Yes	Yes	Yes	**Selected** as Target in Case Study 2
11	Website	Yes	Yes	Yes	**Selected** as Target in Case Study 2
12	Website	Yes	Yes	Yes	**Selected** as Target in Case Study 2
13	Website	Yes	Yes	Yes	**Selected** as Target in Case Study 2
14	Website	Yes	Yes	Yes	**Selected** as Target in Case Study 2
15	Website	Yes	Yes	Yes	**Selected** as Target in Case Study 2
16	File	Yes	Yes	No	**Not Selected** because of unstable operation **
17	File	No	Yes	Yes	**Not Selected**, described as static analyzer #
18	File	Yes	Yes	No	**Not Selected** because of unstable operation **
19	Website	Yes	No	Yes	**Not Selected** because of analysis report with no details ***
20	Website	Yes	No	No	**Not Selected** because of analysis report with no details ***
21	Website	Yes	No	Yes	**Not Selected** because of analysis report with no details ***
22	Website	Yes	Yes	No	**Not Selected** because of unstable operation **
23	Website	No	Yes	Yes	**Not Selected** as it never accessed to-be-tested URLs *
24	Website	No	Yes	Yes	**Not Selected** as it never accessed to-be-tested URLs *
25	Website	No	No	Yes	**Not Selected** as it never accessed to-be-tested URLs *
26	Website	No	Yes	Yes	**Not Selected** as it never accessed to-be-tested URLs *
27	Website	No	No	Yes	**Not Selected** as it never accessed to-be-tested URLs *
28	Website	No	No	Yes	**Not Selected** as it never accessed to-be-tested URLs *
29	Website	No	No	Yes	**Not Selected** as it never accessed to-be-tested URLs *
30	Website	No	No	Yes	**Not Selected** as it never accessed to-be-tested URLs *
31	Website	No	No	Yes	**Not Selected** as it never accessed to-be-tested URLs *
32	Website	No	Yes	Yes	**Not Selected** as it never accessed to-be-tested URLs *
33	Website	No	No	Yes	**Not Selected** as it never accessed to-be-tested URLs *

* We did not observe any access to the to-be-tested URL from these website analysis systems before they provided an analysis report. As a public MSAS-W must access the submitted URL by definition, we immediately screened out these systems. We believe these website analysis systems utilize accumulated database and web reputations for deciding the maliciousness of the submitted URL.

** We could not constantly obtain analysis reports from these systems during the experimental period.

*** These systems provide no detailed analysis reports for us to confirm the effectiveness of NSDI. They only provide their decision whether the submitted URL is malicious or not.

We determined that the system performs not sandbox analysis but static code analysis from the system description and obtained analysis reports.

(Received May 21, 2010)
(Accepted November 5, 2010)
(Original version of this article can be found in the Journal of Information Processing Vol.19, pp.153–168.)

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 Assistant Professor at the Interdisciplinary Research Center, 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.

Yoshihiko Hosobuchi received his B.E. and M.E. degrees in Computer Engineering from Yokohama National University in 2008 and 2010, respectively. His research interest covers a wide area of network security including malware analysis.

Tatsunori Orii received his B.E. in Electrical and Computer Engineering from Yokohama National University in 2009. He is currently a master course student at the Graduate School of Environment and Information Sciences, Yokohama National University. His research interest covers a wide area of network security including malware analysis.

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, an Advisor of Research Center for Information Security, National Institute of Advanced Industrial Science and Technology, 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.