

# Sandbox Analysis with Controlled Internet Connection for Observing Temporal Changes of Malware Behavior

Katsunari Yoshioka, Takahiro Kasama, and Tsutomu Matsumoto

Yokohama National University,  
79-7 Tokiwadai, Hodogaya, Yokohama, Kanagawa 240-8501, Japan  
yoshioka@ynu.ac.jp

**Abstract.** Malware sandbox analysis, in which a malware sample is actually executed in a testing environment (i.e., a sandbox) to observe its behavior, is one of the promising approaches to tackling the emerging threats of exploding malware. One of the important issues of the malware sandbox analysis is that the single execution of the malware sample can reveal only a portion of its potential behavior. Especially, because the recent malware communicate with remote hosts in the Internet for receiving commands and updates, behavior of malware can be diverse depending on those of remote hosts. Another important issue is that we must prevent the executed sample or its attacks from exiting the sandbox if we need to connect the sandbox to the Internet. In order to address the above issues, we propose a totally automated multi-pass sandbox analysis with a controlled Internet connection. In the proposed method, we first execute the sample in an isolated sandbox with the emulated Internet. After the first pass, all outbound connections from the infected host are closely inspected. If the inspections conclude that a connection is harmless, then the sandbox changes the filtering rules to allow it to the real Internet and conducts a new analysis pass. We iterate the above process until no new connections are observed. We evaluated the proposed method with malware samples recently captured in the wild. Applying a simple containment policy, we were able to observe a greater variety of behavior compared with the completely isolated sandbox. Moreover, we conducted the same experiment 5 months after the first experiment. Our findings are as follows; (1) analysis results can differ significantly over time depending on the availability/behavior of remote hosts in the Internet (2) freshness of samples does not always indicate better visibility of their behavior (3) some samples change their behavior depending on the time when they are being executed. These findings imply the importance of follow-up analysis even after samples are captured.

**Keywords:** malware sandbox analysis, Multi-pass Analysis

## 1. Introduction

As the Internet has become a worldwide communication medium in our business and private lives, security threats caused by *malware*, which is a term to indicate malicious software such as computer viruses, worms, bots, Trojan horses, and spyware, have also become critical issues that significantly affect our lives. Consequently great research efforts have been made to tackle them. Sandbox analysis [2, 3, 6, 7, 9, 10, 11, 12, 13, 16] is one of the promising approaches to analyzing malware executables. Its basic idea is to actually execute a captured malware sample in a testing environment (i.e., a sandbox) to observe and analyze its behavior. The observed behavior can be used for conducting various countermeasures, typically for writing a detection signature or a removal tool. One of the advantages of 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 a highly automated fashion.

However, there are several major issues with sandbox analysis. One of them is that the single execution of the malware sample can reveal only a portion of its potential behavior. Especially, because the recent malware communicate with remote hosts in the Internet for receiving command and control (C&C) and updating themselves, etc, behavior of malware can be diverse depending on those of remote hosts. Another issue is that we must prevent the executed sample or its attacks from exiting the sandbox if the sandbox is connected to the Internet.

In this paper, we propose a totally automated multi-pass sandbox analysis with a controlled Internet connection. In the proposed method, we start our analysis with a sandbox that is only connected to an emulated Internet that consists of many dummy servers and hosts with emulated and/or real vulnerable services, called Honeypots in the Sandbox (HitS). Then, all observed connections from the infected host are closely inspected to see if they contain anything harmful. We then change the filtering rules so as to allow harmless connections access to the real Internet and start a next analysis pass. We iterate the above process until no new connections are observed within a threshold number of passes. Advantages of our method over the conventional on-the-fly traffic filtering are as follows: (1) we can test a suspicious outbound connection with HitS and see if it actually compromises it or not while on-the-fly filtering must detect the attack before it actually does any harm to a remote host; and (2) our inspection does not require real time processing as the analysis is done in an iterative way. We evaluated the proposed method with samples recently captured in the wild. By using a low risk containment policy of “allowing only harmless DNS, HTTP and IRC connections to exit,” we were able to observe a greater variety of behavior compared with the completely isolated sandbox with only an emulated Internet. Moreover, we conducted the same experiment 5 months after the first experiment. Our findings are as follows; (1) analysis results can differ significantly over time depending on the availability/behavior of remote hosts in the Internet (2) freshness of samples does not always indicate better visibility of their behavior (3) some samples change their behavior depending on the time when they are being executed.

The rest of this paper is organized as follows: In Sect. 2, we describe related works. In Sect. 3, we explain the criteria of malware sandbox analysis and the proposed sandbox analysis method. In Sect. 4, we explain the experiments for the evaluation of the proposed method. Sect. 5 discusses the limitation of the proposed method. In Sect. 6, we provide conclusions and future works.

## **2. Related Works**

Malware sandbox analysis has been studied intensively in recent years. Previously studies of the malware sandbox analysis can be categorized into two approaches in terms of Internet connectivity; one is a completely isolated sandbox and the other is a sandbox with a real Internet connection. An example of the former approach is Norman Sandbox [13], which emulates the network environment and does not allow any connection to the real Internet. Norman Sandbox emulates many network services such as HTTP, FTP, SMTP, DNS, IRC, and P2P. Other previous studies [6, 7] also take a similar approach.

The limitation of this approach is that it is difficult to emulate remote hosts in the real Internet, as malware produce various kinds of communications. Especially when they talk to one of their servers, such as a C&C server and a file server, they could use arbitrary (even customized) protocols for data transmission and authentication, which makes the emulation increasingly challenging. The other approach mentioned is to carefully connect the sandbox to the real Internet. Examples of Internet-connected sandboxes are CWSandbox [10, 11], Anubis [12], Joebox [16], and [2]. In these literatures regarding such sandboxes, the containment issue, namely, how to filter outgoing attacks, is not concretely discussed. Moreover, the influence on their analysis results from the behavior of remote hosts over time is not concretely discussed either.

Another related technology aiming at a similar goal is a honeypot or a honeyfarm. In [8], outbound traffic from a honeyfarm is analyzed at the application-level to ensure that only C&C communications exit the farm. However, their policy is customized for only certain types of IRC-based C&C communications. One honeyfarm system, Potemkin[9], takes an interesting approach of reflecting outgoing attacks to other virtual honeypot in the farm although the reflecting policies are not concretely presented in the literature. In short, according to our best knowledge, the containment issue of outbound traffic in malware sandbox analysis is still an unsolved critical problem.

### **3. Multi-pass Malware Sandbox Analysis with a Controlled Internet Connection**

#### **3.1 Criteria**

First, we show three criteria of malware sandbox analysis in our consideration.

- Observability
- Containment
- Efficiency

Observability is a criterion of a sandbox analysis in terms of observing malware behavior in consideration. Those who are writing malware removal tools or AV signature may focus on the internal behavior such as changes of registry keys and creation and deletion of files. Network administrators may be interested in their network behavior for writing an IDS signature. In any case, sandbox analysis should be able to provide sufficient information to the analyst.

Containment is a criterion of a sandbox analysis in terms of preventing the analyzed sample from attacking or infecting a remote host outside the sandbox.

Finally, efficiency is a criterion of a sandbox analysis in terms of constantly providing analysis results with sufficient information in a reasonable amount of time. Automation is a big advantage of sandbox analysis compared to other analysis methods such as static code analysis.

The above three criteria naturally have trade-offs in between. For example, one extreme policy to achieve the highest level of containment is to totally isolate the sandbox from outside networks though it would greatly decrease observability. Our focus in this paper is to develop a method that achieves a sufficient level of observability and containment with automation capabilities for retaining the efficiency.

### 3.2 The Proposed Method

We propose a multi-pass sandbox analysis method with controlled Internet connection. We first show an overview of the proposed sandbox method in Fig. 1. Here, the solid lines indicate the communications by the analyzed malware sample and the dotted lines indicate communications by the sandbox system for its operation. The proposed sandbox consists of four components: victim host, Internet emulator, access controller, and analysis manager. The implementation of each component is described in Sect. 3.3.1.

**Victim Host.** The victim host is a host on which a malware sample is first 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. For the experiment explained in Sect. 4, we used Windows XP Professional SP1 as the victim host. It is also important to be able to refresh the operation system (OS) image instantly as it basically gets infected by the malware sample in every analysis pass. Therefore, we used virtualization technology although it is also possible to develop a real machine-based system.

**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 to deal with possibly harmful traffic from the malware that cannot be connected to the real Internet. The Internet emulator consists of various dummy servers such as HTTP, SMTP, NTP, IRC, and DNS servers. In addition to those servers, it also deploys hosts with unpatched vulnerable services called a Honeypots-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 images of the victim host, boots up and shuts down the victim host, executes malware samples in the victim host, receives and inspects all traffic logs and internal logs, generates the filtering rules for the access controller according to the inspection results, and finally outputs the analysis results to the analyst.

**Analysis Procedure.** The following is a brief description of the procedures for the proposed sandbox analysis. We call the sequence of processes from Step 2 to Step 7 an *analysis pass*.

1. The analyst inputs a malware sample and the initial configurations (OS image of the victim host, initial filtering rules for access controller, etc) to the system.
2. The analysis manager reflects the filtering rules to the access controller 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.
4. The access controller redirects the traffic to either the real Internet or the Internet

emulator according to the filtering rules. The incoming traffic to the victim host is not filtered at all.

5. After a certain predetermined amount of 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.
6. The analysis manager receives all logs from the victim host and the Internet emulator and shuts down and refreshes the victim host.
7. The analysis manager inspects the obtained logs and generates the filtering rules. How rules are generated is described in Sect. 3.3.2.
8. If the filtering rules remain unchanged for consecutive  $Th_m$  passes, the analysis manager terminates the entire analysis process. Otherwise, the manager repeats an analysis pass by going back to Step 2.

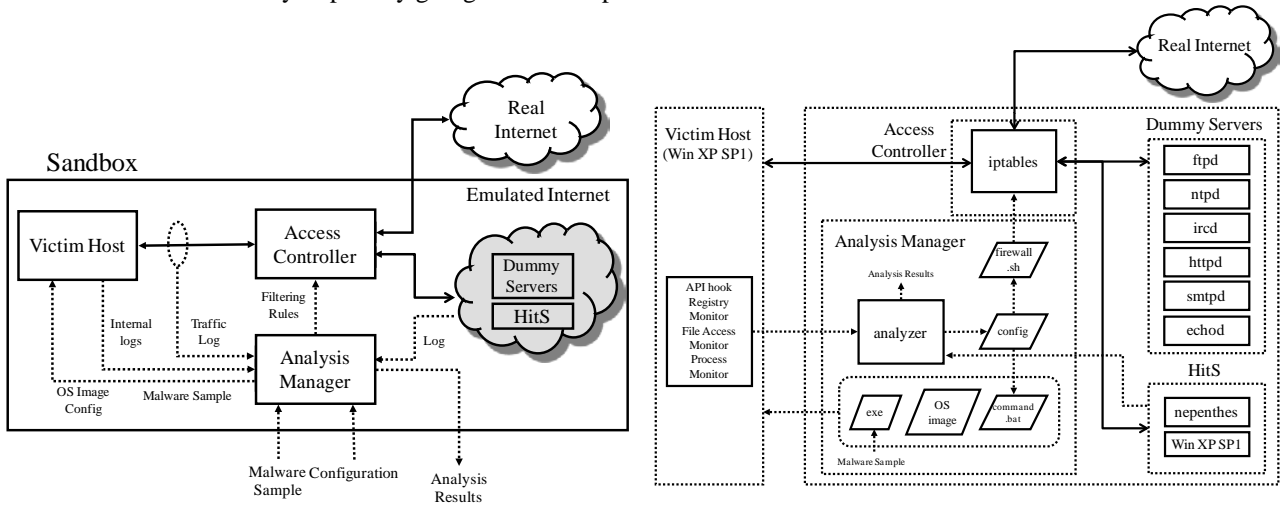


Fig.1. Overview of the Proposed System

Fig.2. Implementation of the Proposed System

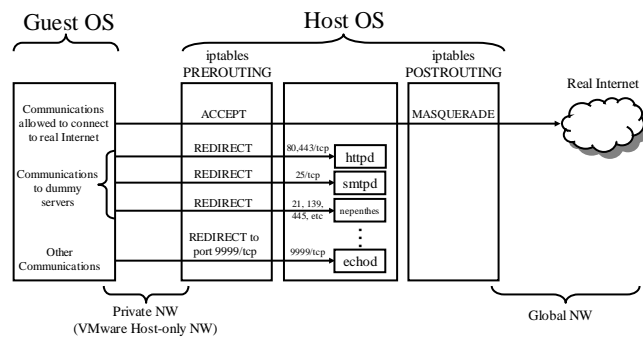


Fig. 3 Outbound traffic filtering by the access controller

### 3.3 Implementation

We show an overview of the implementation of the proposed sandbox analysis system in Fig. 2. Here, the solid lines indicate the communication by the analyzed malware and the dotted line indicates communication by the sandbox system for its operation. The entire system is instantly configured by a single config file.

#### 3.3.1 Components

We implemented the entire system in a single real machine. We used a virtual machine using VMware Server 1.0.6 with Windows XP Professional SP1 running, as the victim host. The host OS is CentOS 5.2, on which the analysis manager, the access controller, and the Internet emulator are implemented. The network between the victim host and the host OS is realized by a virtual private network provided by VMware Server. 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. However, we did not go deep into camouflaging VMware this time such as concealing the VMware I/O backdoor. Presently, basic monitoring tools such as Regmon [17] and Filemon [17] are installed in the victim host. We can also deploy other monitoring tools such as InCtrl [18] or techniques like API hooking [5]. Also, the victim host is configured to automatically download and execute the Windows batch file `command.bat` from the analysis manager upon each boot-up using SSH. The batch file contains these further instructions to be followed by the victim host: (1) downloading the malware sample, (2) starting designated monitoring tools, (3) executing the sample, and (4) sending the monitoring results to the manager after a specified 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, and ECHO servers are implemented as light-weighted simplified server scripts by Perl to emulate the network services in the real Internet. On the other hand, HitS is dedicated to inspecting the connections initiated by the sample to see if they contain any harmful attacks. In order to do this, HitS is designed to run emulated and/or real vulnerable network services for the sample to exploit, like a honeypot. Our current implementation utilizes a low-interaction honeypot program `Nepenthes v0.2.2` [1] as it can emulate multiple vulnerable services. We could also deploy a virtual and/or real machine running a full vulnerable OS to detect zero-day exploits, although such an implementation is our future work. As the dummy servers and HitS listen on only certain ports, we also have the ECHO server, which simply echoes back what it received previously, to reply to connections on any other ports.

**Access Controller.** The access controller is implemented by `iptables`, a packet filtering application program available for the Linux kernels 2.4 and 2.6. Before every analysis pass, a shell script called `firewall.sh` which is the filtering rules generated by the analysis manager, is applied to the access controller. All traffic from the victim host is redirected to either the Internet emulator or the real Internet according to `firewall.sh`. For the

connections to the real Internet, we use the MASQUERADE target of POSTROUTING chain of the iptables. On the other hand, for the connections to the Internet emulator, we use the REDIRECT target of the PREROUTING chain in iptables. We illustrate the outbound traffic filtering by the access controller in Fig. 3. Note that all inbound traffic to the victim host is not filtered at all.

**Analysis Manager.** The process of the analysis manager is described in Sect. 3.3.2.

### 3.3.2 Outbound Traffic Inspection for Containment

In this section, we explain the containment of outbound attacks. The biggest role of the analysis manager is to inspect outbound traffic to decide which sessions are to be connected to the real Internet. According to the inspection results, the manager outputs the shell script `firewall.sh` to be applied to the access controller.

**Statistical Inspection.** The manager inspects all traffic logs obtained between the victim host and the access controller and checks if they contain any port scan or DoS-like behavior from a statistical point of view. 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. We check the DNS name resolution because names of the scanned hosts are normally not resolved by DNS.

**Session-based Inspection.** The analysis manager also performs a dedicated inspection on each TCP and UDP session. Here, a session means a series of packets exchanged between a port of the victim host and a port of a remote host. Each reconstructed session is closely inspected by several tests: an IP spoof test, a destination randomness test, a signature-matching test, and an application protocol test. The IP spoof test simply examines whether the source IP addresses of outgoing packets are spoofed or not. The destination randomness test examines whether the destination IP addresses of outbound connections has been randomly decided on every execution. We consider an outbound connection is suspicious and thus should be filtered, if its destination IP address is randomly decided, because it is a sign of further propagation. In order to check the randomness, the analysis manager first checks if the IP address has been resolved by DNS. If so, we know that the address is not randomly generated. Moreover, the analysis manager records all destination IP addresses that have been targeted by the victim host in previous passes. If an address is visited on every execution, we consider that address hard-coded and, therefore, not randomly decided. The signature-matching test matches the payload of each session with IDS signatures to see if any known exploits are contained within or not. In the present study, we used only a portion of VRT certified rules for Snort v2.8 [14]. Finally, an application protocol test is employed to find out which application protocol is being used in each session. The recognition of application protocols 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.

**Honeypot-based Inspection.** We also perform another inspection using HitS. We check if the vulnerable services of HitS have been exploited by the sample. If any exploitation is confirmed, we do not allow the corresponding sessions to connect to the real Internet. As HitS are currently realized by Nepenthes, we check the successful exploitations by the logs kept by Nepenthes.

**Filtering Rule Generation.** By utilizing the above inspections, the analysis manager derives filtering rules to be applied to the next analysis. The filtering rule generation is carried out as follows:

1. (Statistical Inspection) 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 part of a DoS attack by the inspection.
2. (Session-based Inspection) For each of TCP and UDP session initiated by the victim host, do as follows:
  - 2-1. Do not allow any session whose source IP address is spoofed.
  - 2-2. Do not allow any session whose destination IP address is determined to be randomly generated by the inspection.
  - 2-3. Do not allow any session whose payload contains exploits according to the inspection.
  - 2-4. Do not allow any session whose application protocol is not authorized.
  - 2-5. (Honeypot-based Inspection) Do not allow any session that caused a successful exploitation of the vulnerabilities in HitS.
  - 2-6. Change filtering rules to allow a pair of destination IP address and port from sessions that passed all above inspections.

In Step 2-4, we need to specify a set of authorized application protocols. Presently, rules for ICMP packets are statically given to the analysis manager.

## 4. Experiment

We conducted experiments to evaluate the proposed method using 96 malware samples captured in the wild by Nepenthes v0.2.0 from June 2008 to July 2008. Table 1 is a list of the names and the families of the samples as determined by an online scan at Virustotal.com [15] on May 20th 2009.

### 4.1 Procedures

In order to confirm the observability of the proposed method, we prepared two sandboxes for comparison: the proposed sandbox with controlled Internet access and an isolated sandbox with only the Internet emulator. We will refer to the former system as the *proposed sandbox* and the latter as the *isolated sandbox*. We show the configuration of the proposed sandbox in Table 2. Note that in the proposed sandbox, we only allowed DNS, IRC, and HTTP when connecting to the real Internet after they are being inspected. As a default, ICMP packets are not allowed to exit the sandbox. Each sandbox is implemented on a single real machine with the same specifications: Intel Core 2 Duo Processor 1.20 GHz with a main memory of 2 GB. In order to investigate the changes of malware behavior over time, we conducted the same experiment in Dec 2008 and May 2009. As explained, the proposed sandbox conducts a multi-pass analysis. In the

experiment, the average number of executions per a sample was 3.8 in Dec 2008 and 3.5 in May 2009. Therefore, we conducted performed an analysis using the isolated sandbox for four times independently.

Table 1 Families/Names of Samples obtained from Virustotal [15]

McAfee	Samples	Symantec	Samples
W32/Virut.a	36	W32.Spybot.Worm	22
W32/Bobax Family	20	W32.Virut family	15
W32/Sdbot Family	10	W32.IRCbot family	7
Generic.dx	5	W32.Bobaxldr	5
Generic BackDoor	3	Backdoor.IRC.Bot	2
Trojan.Crypt. Family	3	W32.Linkbot.M	2
Backdoor:Win32/Poebot.AL	1	Infostealer	1
BackDoor-AWQ	1	W32.Rinbot.V	1
New Malware.dq	1	No Information	41
W32/IRCbot.gen.a	1		
W32/Nirbot.worm	1		
W32/Trats	1		
No Information	13		

Table 2 Configuration of Proposed Sandbox

Execution time	30 [sec]
Victim Host OS	Windows XP Professional SP1
HitS	Nepenthes v0.2.2*
Dummy Servers	HTTP (80,443/TCP) NTP(123/TCP, UDP) IRC (6667-6669/TCP) SMTP (25/TCP) ECHO (other ports)
Thresholds	$T_m = 2$ , $T_{ps} = 10$ , $T_{DoS} = 20$
Authorized Application	DNS, IRC, HTTP

## 4.2 Results

Table 3 shows the comparison between the analysis results of the proposed sandbox and the isolated sandbox obtained in Dec 2008 and May 2009, respectively.

**Comparison between Proposed Sandbox and Isolated Sandbox.** In Table 3, average number of destination ports used by a sample, average number of queried domain names, and average size of traffic log all indicate an improved observability of malware behavior by the proposed sandbox. In both sandboxes, more than 70 samples communicated to remote hosts using TCP although the contents of the communication differed significantly between them. There were more than 46 samples that used HTTP GET request in the proposed sandbox while only 22 did in the isolated sandbox. From these requests, over 28 samples received a Windows executable file in the proposed sandbox while none did in the isolated sandbox. As downloaded executables can be used for further malicious activities, it is important to observe such a behavior. Another remarkable difference is the IRC communications. Although IRC NICK or JOIN, used in the beginning point of an IRC session, were observed in both sandboxes, IRC commands like MODE, USERHOST, PRIVMSG, PING were only observed in the proposed sandbox. As PRIVMSG is often used as a C&C command between a bot and its herder, it is important to observe such commands. In fact, many of these PRIVMSG contained a C&C command to start port scan, which were also only observed in the proposed sandbox. There was no noticeable difference in the usage of SMTP as we did not allow SMTP to connect to the Internet.

\* HitS (Nepenthes v0.2.2) is configured to listen on TCP ports 21, 42, 110, 135, 139, 143, 220, 445, 465, 993, 995, 1023, 1025, 2105, 3372, 5000, 10000, and 17300.

### Comparison between Two Periods

Now, we compare the analysis results of the proposed sandbox obtained in Dec 2008 and May 2009. As for HTTP, 39 samples downloaded an executable using HTTP GET in Dec 2008 while 28 did in May 2009. The number decreased because some of the file servers that were available in Dec 2008 were not available or did not return the requested file in May 2009. Such an “expiry” is understandable in the sense that the samples were captured almost a year ago (June-July 2008). Contrastingly, the number of samples that used IRC increased impressively from 37 to 61. Among the 61 samples that were able to connect to their IRC server in May 2009, 27 samples were not able to do so in Dec 2009. We investigated the reasons why these 27 samples became able to connect to their IRC server. Table 4 summarizes the results of the investigation. The following explains each case in Table 4.

Table 3 Comparison of Proposed Sandbox and Isolated Sandbox

	Dec-08		May-09		unit
	Proposed	Isolated	Proposed	Isolated	
Destination ports used per sample	<b>11.9</b>	2.4	<b>4</b>	2.9	(port)
Queried domain names per sample	<b>12.3</b>	9.3	<b>11.5</b>	10.2	(domain)
Size of traffic log per sample	<b>477</b>	33	<b>549</b>	34	(KB)
Analysis time per sample	<b>11.1</b>	12.1	<b>9.7</b>	12.1	(min)
Analysis passes per sample	<b>3.8</b>	4	<b>3.5</b>	4	(pass)
Samples that used TCP	<b>71</b>	74	<b>72</b>	77	(sample)
Samples that used HTTP	<b>46</b>	22	<b>50</b>	22	(sample)
Samples that used HTTP GET	<b>46</b>	22	<b>50</b>	22	(sample)
Samples that received EXE by GET	<b>39</b>	0	<b>28</b>	0	(sample)
Samples that used SMTP	<b>22</b>	22	<b>22</b>	22	(sample)
Samples that used IRC	<b>37</b>	14	<b>61</b>	68	(sample)
Samples that used IRC NICK	<b>37</b>	14	<b>61</b>	68	(sample)
Samples that used IRC JOIN	<b>37</b>	2	<b>46</b>	22	(sample)
Samples that used IRC MODE	<b>21</b>	0	<b>28</b>	0	(sample)
Samples that used IRC USERHOST	<b>21</b>	0	<b>28</b>	0	(sample)
Samples that used IRC PRIVMSG	<b>30</b>	0	<b>21</b>	0	(sample)
Samples that used IRC PING	<b>1</b>	0	<b>3</b>	0	(sample)
Samples that used UDP	<b>82</b>	83	<b>95</b>	84	(sample)
Samples that used DNS	<b>81</b>	82	<b>82</b>	82	(sample)
Samples that used UPnP	<b>22</b>	9	<b>90</b>	32	(sample)
Samples that performed port scans	<b>19</b>	0	<b>20</b>	0	(sample)

Table 4 Investigation on the Recovery of IRC Connections in May 2009

Case	Observation	Possible Reason	Samples
1	Samples used same domain name and IP address, but server responded only in May 2009	Server activated	14
2	Samples used same domain name and IP address but server responded differently in Dec 2008 and May 2009	Server behavior changed	2
3	Samples used same domain name but it was resolved only in May 2009	Domain name resolution activated	5
4	Samples used same domain name but different IP addresses were resolved	Domain name resolution changed	1
5	Samples used different domain name	Sample behavior changed	3
6	Samples started communicating only in May 2009	Sample behavior changed	1

(Case 1) The sample resolved the same domain name and received the same IP address in both periods. The server with this IP address did not respond in Dec 2008 but was indeed available in May 2009. This implies the activation of the server.

(Case 2) The sample resolved the same domain name and received the same IP address in both periods. The server indeed acted as an IRC server only in May 2009. This implies the changes of server’s behavior.

(Case 3) The sample tried to resolve the same domain name but it was resolved successfully only in May 2009. This implies the activation of domain name resolution, possibly by an attacker.

(Case 4) The sample resolved the same domain name but different IP address was returned. This implies the changes of domain name resolution, possibly by an attacker.

Note that the above four cases imply the activities of attackers who have access to the configuration of the IRC server and its domain name resolution. It should be noted that such activities can influence the behavior of malware samples significantly as shown in the experiments.

Finally, the following two cases imply that the malware samples themselves may also change their behavior on each execution.

(Case 5) The sample used a different domain name in May 2009 and was able to connect to an IRC server.

(Case 6) The sample did not communicate to any remote host in Dec 2008 but did so in May 2009.

In Case 5, the domain names resolved by the sample seem to have some dependency on the time when it is executed. Indeed, a malware sample is able to obtain time information from system clock, NTP or any other ways and change the domain names to use. A further investigation is our future work.

**Changes in Domain Name Resolution over Time** We now explain the changes over time in the results of domain names resolution by the samples. From the first experiment in Dec 2008, we obtained 170 domain names from the 96 samples. We wrote a script in order to keep resolving these domain names every hour for over two months (from March 13<sup>th</sup> to May 19<sup>th</sup> 2009). Fig. 4 summarizes the number of changes in the domain name resolution. 62 domain names were never resolved successfully and 67 domain names were constantly resolved to the same IP address. However, there were 41 domain names whose corresponding IP address changed over time. Especially, there were four domain names whose address changed more than 200 times. The domain name “proxim.ircgalaxy.pl” and “proxima.ircgalaxy.pl” changed their addresses for 331 and 320 times respectively. They were constantly assigned one or a few global IP addresses of several countries. The domain name “www.he3ns1k.info” changed its address for 246 times although the assigned address was always one of the loopback addresses 127.0.0.1 and 127.0.0.2. The domain name “gmail-smtp-in.l.google.com” changed its assigned address for 1,376 times. This is one of the domain names of the SMTP servers for Gmail, the free email service provided by Google, as some samples used it for sending a spam. Fig. 5 depicts the number of successfully resolved domain names over time. One can confirm that availability of the domains continuously changed over time, which obviously affected the results of our sandbox analysis.

**Comparison with Other Sandbox Analyzers** We also submitted the samples in Dec 2008 to Norman Sandbox [13] and CWSandbox [11] for comparison. Norman Sandbox and CWSandbox accept an online submission of malware samples. Note that the following results of Norman Sandbox and CWSandbox in Table 5 were obtained just by a single execution and in that sense the comparison is not totally fair. Besides that,

their analysis reports summarized the behavior of analyzed malware and may not fully represent what they were able to observe. Nonetheless, we intend to present these results to show the level of observability achieved by the proposed sandbox. As Norman Sandbox does not connect to the real Internet, the network activities described in their analysis reports were limited, especially, in the case of connections to IRC channels, where only four samples were observed according to their report. On the other hand, as CWSandbox does connect to the real Internet, it was able to observe many communications by the samples. The number of samples that connected to IRC channels was 45, greater than in the proposed sandbox in Dec 2008. However, in May 2009, we were able to observe 61 samples that connected to IRC channels. Note that analyses by Norman Sandbox or CWSandbox were conducted when a sample is submitted for the first time. When receiving any duplicated samples, they just send an analysis report from the previous submission. That means their analysis results could be from a submission prior to ours in Dec 2008. Either way, the proposed method achieved the same level of observability with CWSandbox as far as these particular sample sets are concerned.

# of Changes in Address Assignment	# of Domain Names
Never Resolved	62
0	67
1	5
2	10
3	5
4	4
5	2
6	4
8	5
10	1
12	1
246	1
320	1
331	1
1376	1

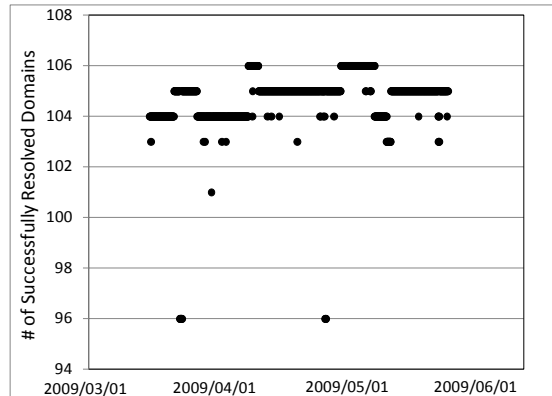


Fig.4 Number of Changes in IP Address Assignment for Domain Names Resolved by Malware Samples

Fig.5 Number of Successfully Resolved Domain Names from March 13<sup>th</sup> to May 19<sup>th</sup> of 2009

Table 5 Comparison with Norman Sandbox and CWSandbox

Sandbox type	Proposed Sandbox (Dec-08)	Proposed Sandbox (May-09)	Norman Sandbox**	CW Sandbox**
# of samples with network activities	71*	72*	23	57
# of samples that connected to IRC channel	37	61	4	45
# of samples that connected to SMTP server	22	22	19	9

\* Number of samples that communicated by IRC or HTTP or SMTP.

\*\* The results of Norman Sandbox and CWSandbox are obtained by a single execution while those of the proposed sandbox are obtained by multi-pass analysis.

**Concrete Analysis Example.** To clearly explain how the proposed sandbox actually works, we show some concrete analysis cases, which are a typical cases of C&C communications and file downloads by a bot. The sample was named

W32/Smalltroj.FHRP (Norton) and Generic BackDoor (McAfee) by virustotal [15] on Dec 11<sup>th</sup>, 2008. The MD5 hash value of the sample is a2cf5b71d9a2589e4c1bef0ece81f910. Note that the IP address and domain name are partly masked by a sequence of a character “x” in the following.

**Analysis Pass 1.** When executed, the sample resolved the domain name “ns.ircstyxx.net” and then tried to access this host (we call it host A) on destination port 1867/TCP using IRC. As this session was not allowed to connect to the real Internet by the initial filtering rules, it was redirected to the ECHO server in the Internet emulator. The sample established a TCP session with the ECHO server but did nothing further. After the inspections, the analysis manager allowed a connection to host A on port 1867/TCP and restarted another pass.

**Analysis Pass 2.** The sample made the same communication as the first pass and tried to connect to host A after domain name resolution. This time the access controller allowed the sample to connect to host A over the real Internet. The sample joined to channel #ns with the name qycayjxx and the nickname CQHRxx. It then changed the IRC mode to invisible and sent the IRC command “USERHOST” to obtain information about the host. After that, the sample tried to connect to another server (we call it host B) on 80/TCP using HTTP after resolving the domain name alwaysssxx.com. As the connection to host B was not allowed, the connection was forwarded to the dummy HTTP server in the Internet emulator. After the TCP session was established the sample requested the transfer of the file lal222.exe to the server using the HTTP GET method. As there is no such file in the dummy server, the session was closed by the sample. After the inspections, a connection to host B was allowed by the analysis manager.

**Analysis Pass 3.** In the third analysis pass, the sample tried to connect to hosts A and B as in the previous pass. This time, it was able to connect to host B over the real Internet and received an executable file. The sample attempted to connect to another server (we call it host C) on 5190/TCP after name resolution. As the connection to host C was not allowed, the session was established with the ECHO server in the Internet emulator. The sample did not show any other network behavior afterwards. After the inspection, a connection to host C was allowed by the analysis manager.

**Analysis Pass 4.** In the fourth pass, the sample connected to hosts A and B and obtained an executable file. This time, host A sent the private message ”#ns.:\*.ipscan.s.s.s.s.dcom2.-s” to an IRC channel. After receiving the message, the sample started TCP SYN scans over a class B network on 135/TCP by sequentially increasing the target IP addresses such as x.x.6.249, x.x.6.250, x.x.6.251. As 135/TCP was not allowed for the real Internet connection, all scans were redirected to HitS in the Internet emulator. The sample established a TCP session with HitS and exchanged a few messages and then finally an exploit code for DCOM vulnerability was sent from the sample. Besides the sequential scans on 135/TCP, the sample also performed random TCP SYN scans on a certain Class B network on 192 different destination ports. We were not able to conclude the intention of these port scans on unusual destination ports. Nonetheless, we were able to detect all scans and redirected them to the Internet emulator.

## 5. Limitations

Although the proposed sandbox showed a certain level of feasibility in the experiments there are several limitations that need to be discussed:

**Detection of Sandbox by Malware.** The malware samples may take several host-based and network-based strategies to detect the sandbox environment. There are a series of technologies to detect virtual machines [4]. Our sandbox can deploy the camouflage technologies although we did not delve into that this time. They can also deploy network-based detection. For example, they can intensively attempt to attack a host under their control and see if the attack actually reaches the host. Because our sandbox does not allow attacks to exit the sandbox environment, such detection may reveal the existence of the sandbox.

**Detection of Sandbox by Bot Herder.** Bot herders who have full access to C&C servers can also check which IP addresses each of their bots are using to connect to their server. As there are normally only a limited number of IP addresses available for sandbox analysis use compared to the number of samples to be analyzed, these addresses would be frequently seen by the herder. It can also be a clue for the herder to find out the IP addresses used by the sandbox. Similarly any particular information that is unique to the sandbox environment such as Windows serial key may be used to detect the sandbox.

Although the above sandbox detection issues seem problematic, we should consider the overall gain with respect to the defense of the real product system. Because the detection of the sandbox may have false positives in general, malware with such detection mechanisms also suffer from false detection of the sandboxes. For example, if malware is designed to stop their malicious activities upon a detection of a virtual machine environment, then it loses the opportunity to infect virtual machines, whether it is actually a sandbox or not. Moreover, we may be able to take advantage of such detection mechanisms to stop malware activities by intensively leading them to false detections. Likewise, although the deployment of a trigger can disturb sandbox analysis, the malware may also lose the chance to infect by waiting for a trigger. In other words, developing effective malware analysis methodologies can increase the cost for the protection mechanism taken by malware, which gives us a reason to work on them.

## 6. Conclusions

We proposed a multi-pass malware sandbox analysis with a controlled Internet connection. In the proposed method, we iteratively executed the target sample in the sandbox and inspected the harmfulness of the observed outbound traffic. If the inspection passed, the connection was allowed to exit the sandbox to the real Internet in the next iteration. This experiment, using in-the-wild samples, showed an impressive improvement in terms of observability of their behavior compared to the isolated sandbox that was connected to the emulated Internet. Also, we confirmed that the results obtained from the proposed sandbox analysis can differ greatly depending on the availability of remote server the samples tried to access. Our future works include a periodical investigation on the malware behavior and their remote servers in order to

connect the analysis of each malware sample with a bigger picture of the malicious activities behind.

## References

- [1] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. C. Freiling, "The Nepenthes Platform: An Efficient Approach to Collect Malware," 9<sup>th</sup> International Symposium on Recent Advances in Intrusion Detection (RAID 2006), pp. 165 - 184, 2006.
- [2] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario, "Automated Classification and Analysis of Internet Malware," Proc. of Recent Advances in Intrusion Detection, RAID07, LNCS Vol. 4637, pp. 178-197, 2007.
- [3] U. Bayer, C. Kruegel, and E. Kirda, "TTAnalyze: A Tool for Analyzing Malware," 15th Annual Conference of the European Institute for Computer Antivirus Research (EICAR), 2006.
- [4] X. Chen, J. Andersen, Z. M. Mao, M. Bailey, J. Nazario, "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.
- [5] H. Father, "Hooking Windows API – Technics of Hooking API Functions on Windows," CodeBreakers Journal, Vol. 1, No. 2, 2004.
- [6] D. Inoue, K. Yoshioka, M. Eto, Y. Hoshizawa, K. Nakao, "Malware Behavior Analysis in Isolated Miniature Network for Revealing Malware's Network Activity," IEEE International Conference on Communications (ICC 2008), pp. 1715-1721, 2008.
- [7] S. Miwa, T. Miyachi, M. Eto, M. Yoshizumi, and Y. Shinoda, "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.
- [8] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "A multifaceted approach to understanding the botnet phenomenon," Proc. 6th ACM SIGCOMM conference on Internet Measurement, pp 41 - 52, 2006.
- [9] M. Vrable, J. Ma, J. Chen, D. Moore, E. Vandekieft, A. C. Snoeren, G. M. Voelker, S. Savage, "Scalability, fidelity, and containment in the potemkin virtual honeyfarm," ACM SIGOPS Operating Systems Review, Vol 39, No. 5, pp. 148 - 162, 2005.
- [10] C. Willems, T. Holz, and F. Freiling, "Toward Automated Dynamic Malware Analysis Using CWSandbox," Security & Privacy Magazine, IEEE, Volume 5, Issue 2, pp. 32 - 39, 2007.
- [11] CWSandbox, <http://www.cwsandbox.org/>
- [12] Anubis, <http://analysis.seclab.tuwien.ac.at/>.
- [13] NORMAN Sandbox Information Center, <http://www.norman.com/microsites/nsic/>
- [14] Sourcefire Vulnerability Research Team (VRT) Certified Rules, <http://www.snort.org/vrt/>
- [15] virustotal, <http://www.virustotal.com/>
- [16] Joebox, <http://www.joebox.org/>.
- [17] FileMon and RegMon for Windows, <http://technet.microsoft.com/en-us/sysinternals/>
- [18] InCTRL Reporting & Analysis, Measuretronix Ltd. <http://www.measuretronix.com/en/products/inctrl-reporting-analysis>