Automated Malware Analysis System and Its Sandbox for Revealing Malware’s Internal and External Activities∗

Daisuke INOUE†[a], Katsunari YOSHIOKA††, Masashi ETO†, Members, Yuji HOSHIZAWA†††, Nonmember, and Koji NAKAO†, Member

SUMMARY Malware has been recognized as one of the major security threats in the Internet. Previous researches have mainly focused on malware’s internal activity in a system. However, it is crucial that the malware analysis extracts a malware’s external activity toward the network to correlate with a security incident. We propose a novel way to analyze malware: focus closely on the malware’s external (i.e., network) activity. A malware sample is executed on a sandbox that consists of a real machine as victim and a virtual Internet environment. Since this sandbox environment is totally isolated from the real Internet, the execution of the sample causes no further unwanted propagation. The sandbox is configurable so as to extract specific activity of malware, such as scan behaviors. We implement a fully automated malware analysis system with the sandbox, which enables us to carry out the large-scale malware analysis. We present concrete analysis results that are gained by using the proposed system.

key words: malware, dynamic analysis, sandbox, security incident

1. Introduction

The Internet has faced various security threats ever since it became widespread. Malware — a generic term for computer viruses, worms, trojan horses, spywares, adwares, and bots — is a key threat. The malicious activities of a great variety of malwares are spread all over the Internet and often lead to serious security incidents that can cause significant damages to both infrastructures and end users.

There are two main approaches to fight against malwares: macroscopic and microscopic. The macroscopic approach is based on network monitoring and focuses on grasping the present trends of malicious activities over networks of a wide range. There are a number of ongoing projects over the world and several monitoring systems are already in their operational phase [1]–[11]. The microscopic approach, on the other hand, is focused on analyzing malware executables that are captured by honeypots, etc., and a web crawler. Captured malware executables are then input to several sandboxes to extract their characteristics and behaviors. This paper concentrates on the microscopic approach.

We have been developing the Network Incident Analysis Center for Tactical Emergency Response (nicter) [12], [13], which integrates both the macroscopic and microscopic approaches. The nicter binds the results of both macroscopic and microscopic analysis to obtain much richer information about malware activities.

The nicter is composed of four systems as depicted in Fig. 1, namely, the Macro analysis System (MacS), the Micro analysis System (MicS), the Network and malware enchainning System (NemeSys), and the Incident Handling System (IHS).

The MacS uses distributed sensors to monitor darknet in several universities and corporations. A darknet is a set of globally announced unused IP addresses and using it is a good way to monitor network attacks such as malware’s scans. This is because there is no legitimate host using these addresses, and we can thus consider all incoming traffic as a consequence of some kind of malicious activities (or that of a misconfiguration.) All incoming traffic is input to analyzers to detect incident candidates such as detection of new scan patterns or sudden increase of scans. We call the monitoring method that quietly monitors incoming packets of a darknet black hole monitoring.

Meanwhile, the MicS captures malwares in the wild by utilizing various types of sensors such as honeypots, and a web crawler. Captured malware executables are then input to several sandboxes to extract their characteristics and behaviors.

The NemeSys enchains the phenomena, i.e., incident candidates, and their root causes, i.e., malwares. Once it has been given an attacking host observed in the MacS, the correlation analyzer in the NemeSys outputs a list of malwares that have similar network behavior (i.e., scans) as the host.
By finding the root causes of the observed network attacks, we have a much clearer view of what is happening in the Internet. Finally, the IHS helps the operator to diagnose the results from the above analyses and make an incident report.

1.1 Contributions

This paper presents the MicS, an automated malware analysis system in the nicter, and its sandbox analysis engine. The main contributions of this paper are as follows.

1. **Totally isolated sandbox analysis**: Unlike some existing dynamic malware analysis systems [14]–[16], we construct a sandbox environment that is totally isolated from the real Internet. This isolation prevents unwanted propagation of malwares during their analysis.

2. **Virtual Internet in the sandbox**: Since most of recent malwares need the Internet reachability [23] for further propagation or for sandbox detection, we introduce a virtual Internet in the sandbox called Internet emulator that imitates several dummy servers in a single machine. The Internet emulator can handle unusual port numbers, which are often abused by malwares, as well as the well known port numbers.

3. **Black hole monitoring in the sandbox**: To extract unaltered scan activity of malwares, which is the key to correlate with the results of black hole monitoring from the MacS, we introduce a methodology of black hole monitoring in the sandbox. This method overcomes a dilemma, that is, how to distinguish between the scan and other network accesses, by taking advantage of DNS query of malwares.

4. **Real machine based analysis**: Current sophisticated malwares have anti-virtualization or sandbox detection techniques by means of finding a certain file, process, or registry entry [24]. To avoid the anti-analysis techniques, we develop a victim host in a real machine, where a malware sample is executed. We make the victim host capable of refreshing its infected OS by use of a dual boot mechanism in order to shorten the analysis time.

Based on the above ideas, we implement a fully automated malware analysis system. The system enables us to carry out a large-scale malware analysis.

This paper is organized as follows. In Sect. 2, we explain some related works. In Sect. 3, we explain the MicS. In Sect. 4, we explain the malware sandbox analysis in detail, and Sect. 5 presents various experimental results. Finally, Sect. 6 gives conclusions and the direction of our future work.

2. Related Work

There are mainly two approaches in malware analysis. One is static analysis and the other is dynamic analysis.

The static analysis is a white-box approach in which the target malware sample is disassembled to enable an analyst to understand its detailed functionalities and code structure. In this analysis, the main problem is often how to disassemble the executables because most of the malware codes are obfuscated by great variety of packers (e.g., ASPack, FSG, Petite, PEX, UPX). This analysis is out of the scope of this paper.

The dynamic analysis is, on the other hand, a black-box approach in which the target malware sample is executed in an environment that is designed to closely observe its internal activities in detail, i.e., its file and registry access, loaded DLLs, created processes, and its API call sequence, and external activities, i.e., server access and scan activity of the malware. A drawback of this analysis is that it only observes a single execution path; however, it still can extract a great deal of information about the behavior of the malware.

The automated dynamic malware analysis system, CWSandbox [14], executes malware on a virtual machine and enables fast analysis. The core engine of Anubis [15], TTAnalyze [16], executes malware on a PC emulator called QEMU [17] to observe Windows native system calls and API calls. Norman Sandbox [18] uses a simulated operating system compatible with Windows for executing malware. Instead of a virtual or emulated machine, a real machine is used for malware execution in Capture [19] and Joebox [20]. Though a real machine takes a long time to refresh its infected OS, it is resistant to the detection of virtual or emulated machines by sophisticated malwares.

These analysis tools focus exclusively on the monitor malware’s internal activity in the infected host rather than on their external activity. This is appropriate for host-based security because the internal activity that is observed can be used for detecting malware or for making removable tools. However, as for the purpose of finding the root cause of the network incident, we need to place much importance on the external activity of malwares.

Another issue is that some of these dynamic analysis tools [14]–[16] allow the malware samples to connect or partially connect the real Internet. It potentially can cause security breaches, namely unwanted propagation of the samples, during the analysis.

Contrastively, the sandbox analysis proposed in this paper focuses on analyzing malware’s external activity. Doing this means that the analysis results can be directly linked with the results from the network monitoring. In addition, the sandbox is totally isolated from the real Internet in order for the secure analysis of malwares.

3. MicS: Automated Malware Analysis System

We now explain the overview of our proposed malware analysis system MicS. We first explain three requirements for the architecture of the MicS:

1. **Scalability**: As the nicter collects a large number of malware samples every day, the MicS needs to be able to handle them on time.

2. **Flexibility**: To deal with the diversity of malwares, new analyzers should be easily deployed into the MicS.

3. **Isolation**: As the MicS deals with actual malware
executables, which could potentially be activated for further infection, the analysis environment needs to be totally isolated from outside networks in order to avoid unwanted propagation.

The architecture of MicS is shown in Fig. 2. It consists of several components: importer, exporter, gatekeeper, component manager, Anti Virus (AV) scanner, sandboxes and data analyzer. All malware samples captured by the sensor (shown in Fig. 1) are automatically submitted to the importer. The gatekeeper periodically (e.g., every few seconds) downloads the submitted malwares from the importer. When a new malware is submitted (i.e., “new” in terms of MD5 hash values), the gatekeeper passes it to the component manager to start the analysis.

In the meantime, the gatekeeper sends the sample to the AV scanner to obtain its names (if known). Once the component manager has received the submission, it sends a request to all (or some of) the sandboxes to start analyzing the submitted sample. After finishing the analysis, each sandbox sends the resultant logs to the data analyzer. The data analyzer analyzes the logs and sends the analysis results to the component manager. The component manager sends them to the gatekeeper. Finally, all the analysis results along with their names defined by the AV scanner are output to the exporter. The processes of the MicS are fully automated from the submission of malware samples to the output of analysis results.

In order to enhance scalability, all processes are automated and well-managed by the component manager. The manager monitors the status of each sandbox and dispatches a request for analysis on the basis of their status and load thus ensuring that the submitted malwares can be analyzed efficiently. It is also possible to use multiple sandboxes to enhance the performance.

The parallel processing feature also gives flexibility to the MicS. Whenever we deploy a new analysis engine we simply configure the component manager to dispatch the analysis request to the new engine.

To achieve isolation, we construct the sandbox that includes an emulated Internet environment. In this network, various network services that we anticipated the malwares would use are emulated in order to obtain deeper knowledge of the network activities of malwares. Unlike previously studied dynamic malware analysis systems [14]–[16], the sandbox in the MicS is totally isolated from outside networks and therefore ensures much higher reliability in terms of preventing unwanted propagations outside the analysis system.

4. Sandbox Analysis

As briefly explained in the previous section, the sandbox in the MicS, which is based on the dynamic analysis, focuses not only on internal activity of malwares but also on their external activity.

In order to extract a malware’s internal and external activities, we constructed an analysis environment in which a real machine, called victim host, is infected by the sample malware. The victim host is not connected to the real Internet but to an isolated mimetic network, called an Internet emulator. Since the sandbox is totally isolated from outside networks, it does not cause any unwanted infection.

The API calls in the victim host, server logs in the Internet emulator, and packets transmitted between the victim host and the Internet emulator are collected. All the collected information is then analyzed by the data analyzer and finally high-level descriptions of the observed activities are output in XML format. We illustrate the overview of the sandbox analysis in Fig. 3.

4.1 Victim Host

The victim host is a real machine in which the target malware sample is executed and infected. To extract the internal activity, it hooks and monitors Windows API calls used by the malware by means of an API hook technique that is similar to the conventional one [21].

First, the victim host executes the malware by the createProcess() function with the suspend mode. A remote thread is then created on the malware process. The remote...
thread loads a DLL named API_Hook.Lib.dll. Meanwhile, API addresses on the Import Address Table (IAT) are overwritten so as to redirect API calls to the API_Hook.Lib.dll. In the DLL, API calls are logged and redirected to corresponding original APIs. Even APIs in a DLL that is linked in run-time can be monitored because the LoadLibrary function is also hooked. When the LoadLibrary is called by the malware, the API_Hook.Lib.dll returns a fake handle to lead the malware to load the DLL itself.

After a given period of time, e.g., 30 seconds, the victim host outputs the Windows API logs to the data analyzer. Empirically, most malwares expose a greater part of their activity in 30 seconds. If the malware sleeps during the execution, the victim host forcibly cancels the sleep by means of hooking and skipping the sleep function. If the malware adds run keys in the Windows registry to enable the auto run, the victim host restarts the Windows OS and conducts the analysis again to extract the malware’s activity after the restart. This causes some variations in the analysis time as six to nine minutes per sample (Sect. 5.2).

The victim host is realized by using a Linux/Windows dual boot machine. The machine is configured to boot alternately with either Linux OS and Windows OS by a boot loader. The target malware sample is executed when Windows OS is running. After the malware execution, the Windows OS shuts down automatically and the Linux OS is then booted. To refresh the Windows OS, the pre-stored image of the original Windows OS (before the infection) is written over the infected Windows OS image on the Linux OS. This refresh process, which takes about five minutes, takes up most of the analysis time (Sect. 5.2).

4.2 Internet Emulator

Internet emulator provide a virtual Internet environment to the victim host so that the infection of the victim host proceeds and more detailed behavior of malware can be observed. There are two requirements.

(1) Availability of network services: Recent malwares often use several network services such as DNS, HTTP, and SMTP to propagate over networks. They also use the network services to check the reachability to the Internet. If the malwares can not access the services, they stop their action to avoid the malware analysis in a quarantine environment. Therefore when the victim host requests to access the network services, the Internet emulator needs to respond appropriately.

(2) Observability of scan with no response: Self-propagating malwares perform scans such as a TCP SYN one or an ICMP ping one to find its next target of propagation. These scans are what we can observe using the black hole monitoring in the MacS. However if the internet emulator responds to scan packets, some malwares can change their behavior. An example is that the W32.Pinfi drastically changes its behavior by receiving a SYN-ACK for its SYN scans (Fig. 4). The malware quickly responds an ACK and sends extra packets within 10 msec. This change causes a mismatch between analysis results from MacS and MicS. As a result, the NemeSys (Fig. 1) may not conduct the correlation analysis. Consequently the Internet emulator should not respond to scan packets from the victim host.

In order to satisfy the former requirement, the Internet emulator involves various kinds of dummy servers, such as DNS, FTP, TFTP, HTTP, HTTPS, SMTP and IRC to respond to the victim host. For instance, when a DNS query is issued by the victim host, the query is sent to the dummy DNS server. When a bot in the victim host sends an HTTP request to a certain web server, which can be a honeypot detection [23], the dummy HTTP server can respond arbitrary contents so as to imitate a command from a bot herder.

Here we need to consider that there are many malwares using unusual port numbers to access the network services. Typically, many bots use large port numbers (like 7000, 10324, 65520, etc.) instead of the default port 6665 to 6669 for IRC access. To deal with the unusual port number, the Internet emulator introduces an intermediate module called Dummyd as illustrated in Fig. 5. All packets generated by the malware will be forwarded to the Internet emulator. When the Internet emulator receives a packet on some well-known port numbers such as TCP/20, TCP/21, TCP/25, UDP/53, etc., it is redirected to a corresponding dummy server. Otherwise, all TCP SYN packets are sent
to Dummyd, which simply sends SYN-ACK back. Such packet flow is controlled by the iptables, which is a packet filtering tool on Linux. When the Dummyd receives a first packet after the three-way handshake, it parses its payload in order to check if the protocol is either HTTP or HTTPS or IRC. It then redirects the packet to an appropriate server. If it decides that the protocol is neither HTTP nor HTTPS nor IRC, it simply discards the packet.

For any SYN packets with a well-known port number, the Dummyd always sends back SYN-ACK packets; therefore this is the sandbox with SYN-ACK mode. Although this mode can deal with the unusual port numbers, it does not satisfy the second requirement, that is, observability of scan with no response. If a malware performs a SYN scans toward some unusual port number, the Dummyd responds SYN-ACKs for all scans.

Fulfilling the above two requirements, the most challenging task of the Internet emulator with the black hole mode is to distinguish between scans and the accesses to network services. The fundamental problem is that we do not know if a TCP SYN from the victim host is meant for an access to a certain server or a scan to a randomly decided address unless we respond a SYN-ACK and check the further actions of the victim host. However, if we send a SYN-ACK to a scan packet, it would not be black hole monitoring anymore.

To solve this dilemma, we take advantage of a typical malware activity, that is, DNS query. It is no wonder that most of malwares that access to certain server send DNS query beforehand in order to resolve the IP address of the server. Here we can assume if a packet from a malware goes to an IP address resolved by DNS server, the packet is for the network access to certain server. If not (i.e., the destination IP address has never been resolved by DNS server) the packet is for the scan. In other words, the Internet emulator should respond only to the packets whose destination IP address were resolved by the dummy DNS server.

Figure 6 illustrates a construction of the Internet emulator with black hole mode. The dummy DNS answers an IP address within a certain network range (e.g., x.y.z.0/24) for a DNS query from the malware. Only the packets directed to the network range can through the iptables, and then reach the corresponding servers or the Dummyd according to their port number. Packets directed to outside the network range are regarded as the scan, and thus just filtered. This mode of the Internet emulator satisfies both of the two requirements, namely this is the sandbox with the black hole mode.

### 4.3 Data Analyzer

The data analyzer analyzes all logs sent from the victim host and the Internet emulator. It can translate low-level logs into high-level behavior patterns taking advantage of the behavior pattern database that stores many behavior patterns of malwares. Eventually, an analysis result is created as an XML. Appendix shows an example of the analysis result in an XML format.

### 5. Experimental Results

In this section, we show some experimental results of the MicS. First, the system specifications are described. Second, we measure the analysis time of certain malwares. Third, statistical analysis results taking advantage of the automated dynamic analysis are presented.

#### 5.1 System Specifications

To implement the system illustrated in Fig. 2, we use the machines with the same hardware show in Table 1. The OS information of each machine is shown in Table 2.

#### 5.2 Analysis Time

Through an analysis cycle of a malware, the victim host consumes the processing time dominantly because the procedure on the victim host includes the execution of the malware and a refresh of the Windows. Processing time on
the manager and the data analyzer are negligible in comparison with that on the victim host. Therefore we measure the time of an analysis cycle on the victim host to show the performance of the system. Figure 7 shows an analysis cycle on the victim host. As mentioned in Sect. 4.1, a malware sample is executed once or twice according to the run key related operation. If the malware adds itself to the run key in the Windows registry during the 1st execution, then the Windows OS is restarted and the 2nd execution is carried out. After the two executions, the Linux boots and refreshes the Windows OS image by copying the pre-stored image from the hard disk in the victim host (Fig. 7 (b)). Otherwise, the malware is executed only once, and then the Windows OS image is refreshed (Fig. 7 (a)).

We measure the analysis time of two malwares: W32.Virut.B and W32.SillyFDC (Symantec name). The former malware does add itself to the run key and the latter one does not. The execution time in the 1st and 2nd execution is configured as 30 seconds. The internet emulator is the black hole mode. Table 3 shows the results of the measurements. The total analysis time is about six to nine minutes; hence a single set of the sandbox can handle 150 to 250 malware samples per day. The Windows restart and Windows image refresh consumed about 85 to 90 percent of the analysis time, which is compensation for the use of the real machine as a victim host. To reduce the time of these processes, the use of RAM disk is a promising way. Otherwise, as mentioned in Sect. 3, the system is flexible to parallelize the multiple sandboxes to enhance the capacity of analysis. Currently, the nicter has eight sets of sandboxes with the black hole mode, which can analyze about 1200 to 2000 malware samples per day.

5.3 Statistical Analysis

We now explain the statistical analysis results of the malware sandbox analysis. The following results are derived from 747 malware samples that were captured by an open source honeypot — Nepenthes [22] (ver. 0.2.0) — from Aug. 2007 to Dec. 2007. The malware names by Norton AntiVirus 2007† and McAfee VirusScan for Client†† are shown in Fig. 8. Malware name with the symbol “*” means the wild card of malware variants.

5.3.1 PE Validation Check

At the beginning of the 1st execution, the victim host calls the createProcess() with the suspend mode (Sect. 4.1) to hook and execute a malware. However the malware samples captured by the Nepenthes sometimes include unexecutable files. When the victim host finds that there is no API log after the 1st execution, it again calls the createProcess() with normal mode. If the function returns an error code, the victim host judges and logs the sample is an invalid portable executable (PE) file.

Figure 9 shows the analysis result of the PE validation check. There are 99 (13.3%) invalid PE files out of the 747 samples. Hereafter we use the 648 valid PE files for the population of the analysis.

---

Table 3 Analysis time.

<table>
<thead>
<tr>
<th></th>
<th>W32 Virut.B 1</th>
<th>W32 SillyFDC 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st execution</td>
<td>37.35 sec.</td>
<td>48.55 sec.</td>
</tr>
<tr>
<td>Windows restart</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2nd execution</td>
<td>305.75 sec.</td>
<td>306.25 sec.</td>
</tr>
<tr>
<td>Windows image refresh</td>
<td>343.10 sec.</td>
<td>541.75 sec.</td>
</tr>
</tbody>
</table>

* The results are the mean value of continuous twenty analysis cycles of each malware samples.

1 MD5 value: 58dec21c7e338d0412a791d54e6f4e76
2 MD5 value: 94f8478cd51a54305e5e64d77e28

† Signature versions: 2007.10.08.016 to 2007.12.16.003
†† Signature versions: v5136 to v5186
5.3.2 Number of API Calls

The distribution of the number of API calls in the first execution observed by the API hooking is shown in Fig. 10. Each dot corresponds to a sample where the horizontal axis indicates the number of API calls in the black hole mode and the vertical axis indicates that in the SYN-ACK mode.

In the case of the black hole mode, the largest number of API calls observed by a single execution of a sample was approximately 82,000 (W32.Dabber.B). In the case of the SYN-ACK mode, the largest one was about 156,000 (W32.Gobot.A). The average number of API calls in the black hole and SYN-ACK mode are about 881 and 4,452, respectively, which indicates that the API calls appear more often in the SYN-ACK mode than in the black hole mode. The plots placed over the dashed line indicate that these samples called API more often in the SYN-ACK mode than in the black hole mode, and vice versa.

Eight malware samples did not mark any API logs in both modes. According to our detailed analysis, we found that these samples locates the memory address of APIs and uses them directly instead of using the IAT. It is our future work to hook the API calls of these malwares.

5.3.3 Server Access

The server accesses observed in the SYN-ACK mode and the black hole mode are shown in Fig. 11. The results of the two modes are almost identical. There is no access to NTP server. There is also no access to FTP nor to the TFTP server. We consider that it is because the propagation steps of the executed samples did not quite reach the phase of downloading files. It is notable that more than a quarter of the samples do try to access a DNS and an IRC server. The present version of the Internet emulator emulates an IRC session until the IRC user (i.e., the malware) joins a certain channel. To extract further behavior, we need to emulate the command and control (C&C) messages from the bot herders in the sandbox.

There are several malwares that access to TCP/443, i.e., the default port for HTTPS. They, however, establish an IRC connection via the port instead of establishing an HTTPS connection.

---

†MD5 value: 5bfd3657259a3f26d00f242487037304
††MD5 value: 7fdfe363d51e27caa1b6d490646e66f5
5.3.4 Port Number for IRC Connection

We list all the port numbers that are accessed by the malware samples for IRC connection in Table 4. There is a wide variety of port numbers selected for the IRC connection.

The bold numbers in the table indicate the port numbers that are accessed only in the corresponding mode. Not a few samples seem to choose different port numbers at each execution such as W32.IRCbot† and W32.Zotob.E††.

5.3.5 Scan Activity

In the black hole sandbox, scans and accesses to network services issued by malwares could be correctly distinguished by means of the coordinated movements of the dummy DNS and the packet filtering with the iptables. Thus the black hole sandbox could provide unadulterated scan activities of malwares, which is the key to correlate with the results of network monitoring from the MacS.

Out of the 648 samples, there were only 38 (5.9%) samples that performed global scans in the both mode. These samples mostly sent SYN packets on the ports 139, 445, 3127 and ICMP echo requests. Of the other 610 samples that did not perform global scans, 174 accessed an IRC server and stopped their activities afterwards. As we consider them to be bots, our next approach is feeding them C&C messages by emulating the behavior of the herder. The scan activity of the examined samples is shown in Fig. 12.

<table>
<thead>
<tr>
<th>Table 4</th>
<th>Port list for IRC connection.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Port List</strong></td>
<td></td>
</tr>
<tr>
<td>SYN-ACK Mode</td>
<td>1814, 1863, 2234, 3938, 4545, 5190, 5900, 6667, 6668, 6697, 7000, 7654, 7763, 7878, 8080, 8998, 9283, 9889, 10324, 11000, 31090, 65146, 65520</td>
</tr>
<tr>
<td>Black Hole Mode</td>
<td>443, 1814, 2020, 2234, 3938, 5554, 5900, 6659, 6667, 6668, 6697, 7000, 7654, 7878, 8080, 9928, 11000, 31090, 51115, 65146, 65520</td>
</tr>
</tbody>
</table>

![Fig. 12](Image) Scan activity.

6. Conclusion

We focused on the external activity derived from the malware, and proposed and developed the automated malware analysis system with the isolated sandbox environment. The sandbox has the following novel technical features: First, we created a secure environment for the sandbox because the network is totally isolated from the real Internet and, unlike previously proposed methods, the execution of the malware causes no further unwanted propagation. Second, we introduce a black hole sandbox, which utilizes the coordinated movements of the dummy DNS and packet filtering. It is suitable for observing the unadulterated scan activity of malware and that enables efficient binding with the actual scan monitoring over the global network. Third, to correctly analyze the malwares that are capable of detecting even virtual or emulated machines, the sandbox is well constructed by a real machine. We presented the results of our sandbox analysis that demonstrate the capability and performance of the proposed malware analysis system MicS.

Our future work will be to advance the sandbox in order to emulate the C&C message, which enables us to control the bots. One of our goals is an efficient integration of the microscopic and the macroscopic analysis. Secure, large-scale and flexible malware analysis system in an automated manner is a key component to attain the goal.

References

[7] JPCERT/CC Internet Scan Data Acquisition System (ISDAS), http://www.jpcert.or.jp/isdas/


Appendix: Analysis Result of W32.Linkbot.M

The following XML description is a part of the analysis result of the W32.Linkbot.M, which was automatically generated by the data analyzer.

```xml
<MaliciousCodeXML>
  <MaliciousCode>
    <Name>POEBOT.exe</Name>
    <Action>
      <deleteFile>
        <id>00700002</id>
        <time>06:37:12.643000</time>
        <path>C:\WINDOWS\System32\csrs.exe</path>
      </deleteFile>
      <copyFile>
        <id>00200003</id>
        <time>06:37:12.658000</time>
        <file>Source:C:\Victim Emulator\vemu\virus\POEBOT.exe Destination:C:\WINDOWS\System32\csrs.exe</file>
      </copyFile>
      <!-- skip some lines -->
      <addReg>
        <id>00010001</id>
        <time>06:37:19.080000</time>
        <desc>It will automatically run each time Windows is started.</desc>
        <registrykey>HKEY_LOCAL_MACHINE\Software\Microsoft\Windows \CurrentVersion\Run\ClientServerRuntimeProcess</registrykey>
        <registryvalue>C:\WINDOWS\System32\csrs.exe</registryvalue>
      </addReg>
      <!-- skip some lines -->
      <connect>
        <id>06000002</id>
        <time>06:37:20.502000</time>
        <desc> Attempts to connect to server.</desc>
        <IP>88.117.40.45:6667</IP>
      </connect>
      <connect>
        <id>80000030</id>
        <time>06:37:20.951000</time>
        <desc>SMB Protocol.</desc>
        <IP>192.168.20.255:137</IP>
      </connect>
      <!-- skip some lines -->
    </Action>
  </MaliciousCode>
</MaliciousCodeXML>
```
Daisuke Inoue received his B.E. and M.E. degrees in electrical and computer engineering from Yokohama National University in 1998 and 2000, respectively, and Ph.D. degree in engineering from Yokohama National University in 2003. He joined the Communications Research Laboratory (CRL), Japan, in 2003. The CRL was relaunched as the National Institute of Information and Communications Technology (NICT) in 2004, where he is a senior researcher of the Information Security Research Center. His research interests include security and privacy technologies in wired and wireless networks, incident handling based on network monitoring and malware analysis. He received the Best Paper Award at the 2002 Symposium on Cryptography and Information Security (SCIS 2002).

Katsunari Yoshioka received the B.E., M.E. and Ph.D. degrees in Computer Engineering from Yokohama National University in 2000, 2002, and 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 wide range of information security, including malware analysis, network monitoring, intrusion detection, and information hiding. He received the Best Paper Award at the 3rd Joint Workshop on Information Security (JWIS 2008).

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

Yuji Hoshizawa joined Symantec in 1998, took a position in charge of security research, correspondence to new viruses, and collection and analysis of vulnerability information as the Asia Pacific regional manager of the Symantec Security Response. He has established himself as a top class virus researcher in Japan, and has been contributing to many IT related publications about computer security. Moreover, he gave talks at the various international conference such as Virus Bulletin, EICAR, and AVAR, on the subject of security issues. After leaving Symantec in September 2004, he joined SecureBrain Corporation and took a charge of the present position since October 2004.

Koji Nakao is the Information Security Fellow in KDDI, Japan. Since joining KDDI in 1979, he has been engaged in the research on multimedia communications, communication protocol, secure communicating system and information security technology for the telecommunications network. He is also an active member of Japan ISMS user group, which was established in the 1st Quarter of 2004. He is the board member of Japan Information Security Audit Association (JASA) and that of Telecommunication ISAC Japan, and concurrently, a Technical Group Chairs (ICSS: information communication system security) of the Institute of Electronics, Information and Communication Engineers. He received the B.E. degree of Mathematics from Waseda University, in Japan, in 1979. He received the IPSJ Research Award in 1992, METI Ministry Award and KPMG Security Award in 2006, and Contribution Award (Japan ITU), NICT Research Award, Best Paper Award (JWIS) and MIC Bureau Award in 2007. He is a member of IPSJ. He has also been a part-time instructor in Waseda University since 2002.