

# Malware Detection Focusing on Behaviors of Process and its Implementation<sup>\*</sup>

Yoshiro Fukushima<sup>1 2</sup>, Akihiro Sakai<sup>1</sup>, Yoshiaki Hori<sup>1 2</sup>, and Kouichi Sakurai<sup>1 2</sup>

<sup>1</sup> Graduate School of Information Science and Electrical Engineering,  
Kyushu University,  
744 Motoka, Nishi-ku, Fukuoka 819-0395, Japan

<sup>2</sup> Institute of Systems and Information Technologies and Nanotechnologies,  
2-1-22 Momochi-hama, Sawara-ku, Fukuoka 814-0001, Japan

**Abstract.** In recent years, malwares spread faster and become cleverer to evade anti-virus software. So, the capability of anti-virus software is limited to some extent. Recently, the methods based on malicious behaviors of malwares are proposed. However, there is a problem of false positives in these methods. In this paper, we propose a malware detection scheme based on evaluation of suspicious process behaviors on Windows operating system. We consider an event of creating files to operating system and temporary folders as a malicious behavior. And, we consider an event of executing the created file as a malicious behavior. Moreover, to decrease false positives, we consider an event of registering and deleting uninstallation information as a non-malicious behavior. The result of our experiment shows that our proposal detected about 60% of malwares and there were no false positives. Using our tool, we can verify whether a program is malware or not automatically, and reduce the time of analyzing the behavior of programs.

**Key words:** malware detection, malware analysis, dynamic analysis

## 1 Introduction

In recent years, with the development of networks, there are more opportunities for us to receive a wide range of services through the Internet. These services are indispensable for us now.

However, the cases that malicious people damage others' computers through the Internet have been increasing. In particular, the damage caused by malwares such as computer virus, trojan horse, and worm, is huge. It is common to introduce anti-virus software as an anti-malware countermeasure. In the anti-virus software, a signature based method is a main technique. This method uses the characteristic patterns of each malware and detects malwares by comparing these patterns with byte code of a malware. But, this method cannot detect malwares

---

<sup>\*</sup> This research was partly supported by a contract with the National Institute of Information and Communications Technology, entitled "Research and Development for Widespread High-speed Incident Analysis."

such as unknown, self-encrypting, and self-obfuscating malwares because it is hard to create patterns from these malwares.

To address that problem, a behavior based technique which focuses on malicious behaviors of malwares is used. In the behavior based technique, behaviors that seem to be malicious are defined beforehand. If a program behaves in these defined behaviors, it is detected as a malware. However, it is usually hard to define these malicious behaviors accurately, so there is a problem regarding true positives of malwares and false positives of normal programs in the behavior based technique. For example, the false positive rate could be increased when we try to raise the true positive rate, and the true positive rate could be decreased when we try to decrease the false positive rate oppositely.

In this paper, we propose a malware detection scheme based on evaluation of suspicious process behaviors on Windows operating system. In our proposal, we aim to decrease false positives. To begin infection activities, malwares often create execution files such as duplication of themselves to operating system and temporary folders, and execute these files continuously. We define these behaviors as “Creation of files to particular folders” and “Execution of the created files” respectively, and consider these behaviors as malicious. Moreover, to decrease false positives, we also focus on the behavior of “Registration and deletion of uninstallation information” that malwares would usually not do. The result of our experiment shows that our proposal detected about 60% of malwares and there were no false positives. In addition, we implemented our proposal as a tool which can automatically verify whether a program is malware or not. And, by using the tool, we can reduce the time of analyzing the behavior of programs to about 2-3 minutes.

The composition of the remainder of this paper is as follows. In section 2, malware detection techniques are described, and in section 3, we describe related works. In section 4, we describe our proposal, and in section 5, we describe the result of our experiment. And, in section 6, we describe the implementation of our proposal. Finally, we describe the conclusion and future works.

## 2 Malware detection techniques

There are mainly three malware detection techniques as follows. We describe those techniques and features.

### – Pattern matching technique

In this technique, characteristic patterns of malwares are registered beforehand as signatures, and it detects malwares by comparing these patterns with byte code of a malware. Pattern matching technique is effective for existing malwares whose signature has been created. However, this technique cannot detect unknown malwares or subspecies whose signature is not created yet. Moreover, it is hard to create signatures of self-encrypting and self-obfuscating malwares, so it is difficult for this technique to detect those malwares.

- **Heuristic technique**

In this technique, instead of actually executing a program, it disassembles the code of the program and analyzes its disassembled code statically. And, this technique detects malwares whether there are malicious codes in the program. Unlike the pattern matching technique, this technique can detect malwares even if they are unknown malware or subspecies. However, it is hard to define what malicious codes are, so there is a problem of false positive that non-malicious programs are judged as a malware. Moreover, this technique is based on static analysis, so it is hard to detect malwares such as self-encrypting and self-obfuscating malwares.

- **Behavior based technique**

In this technique, it dynamically analyzes behavior of programs by actually executing them, and detects malwares whether the observed behaviors are malicious behaviors. Like the heuristic technique, behavior based technique can detect unknown malwares or subspecies. Moreover, because this technique is based on the actual behavior of malwares, it can detect malwares such as self-encrypting and self-obfuscating malwares. However, like the heuristic technique, it is hard to define malicious behaviors accurately, so there is a problem of false positive. And, for the malwares that act illegally only in a specific condition such as dates, if that condition is not satisfied, the malicious behaviors are not observed, so this technique cannot detect those malwares.

In this paper, we propose the behavior based method which can detect unknown, self-encrypting, and self-obfuscating malwares.

### 3 Related Work

Dynamic analysis is used for analyzing the behavior of programs in the behavior based method. In dynamic analysis, by inspecting system call or API which programs call, it analyzes what the behavior of programs is[2]. In addition, for dynamic analysis, there are other ways to analyze behavior of programs. One is using a monitoring tool which can observe file access, registry access and so on of programs in real time. The other is using analysis systems through Web service such as Anubis[1], CWSandbox[4], and Norman SandBox[5] which send a report about the behavior of the program that we post into these site.

The techniques focusing on the system call and API have been researched for anomaly detection and intrusion detection mainly[8][10]. Each study basically characterizes normal behaviors from the sequence of system call and API calling, or the arguments of these. If observed behaviors deviate from characterized normal behavior, anomaly is detected. These techniques can be applied to malware detection technique. On the other hand, there are some researches which characterize the malicious behaviors from API sequences that malwares call. In these researches, the characterized behaviors are used for malware detection[3][6][7][11]. However, those researches mentioned above have to learn normal

or malicious behaviors in advance. Our proposal need not to learn these behaviors in advance because our proposal detects malware whether specific behaviors are observed.

## 4 Proposal

In this paper, we propose the behavior based scheme. We describe the behaviors which are focused on our proposal below.

### 4.1 Creation of files to particular folders

To begin infection activities, malwares often create their duplication and other malicious files to particular folders. We define these particular folders as following three folders.

- **C:\Windows\System32 (System)**  
In the following, we call this folder “System folder”.
- **C:\Windows**  
In the following, we call this folder “Windows folder”.
- **C:\Documents and Settings\User\Local Settings\Temp**  
In the following, we call this folder “Temp folder”.

Because various files exist together in these folders, creating malicious files are not noticed easily by the user. Hence, it is thought that malwares create files to these folders mainly.

### 4.2 Execution of the created files

In normal programs, there is a case that files are created to the particular folders mentioned above. For example, temporary files to save working contents are often created to the Temp folder. However, in normal programs, the kind of executing the created files is limited to some extent. As the instance of such normal programs, there are installers and uninstallers. An Installer is a system for installing software on your PC and an uninstaller is one for removing software from your PC. In addition, we consider the following two baselines for judging whether it is executed.

- 1 . **Execute until process created by execution of a program is exited (call “executing immediately”)**
- 2 . **Execute until next boot of Operating System**

About the second baseline, we consider a case that malwares set up system so that they are executed automatically at the time of OS start. In that case, although the created files are not actually executed, by extending the baseline of the execution to this, it is thought that true positive of malwares can be improved. The reason is that malwares often set up system so that they are executed automatically at the time of OS start to infect thoroughly to system. Hence, we decided to consider also the second baseline. There are two techniques for executing oneself automatically at the time of OS start: registration to “Run registry” or “Start-up”.

We did the preliminary experiment in advance to analyze the behaviors of some malwares and normal programs. As a result, we found that there was a problem that non-malicious installers and uninstallers caused false positive when focusing on the two behaviors mentioned above (“Creation of files to particular folders” and “Execution of the created files”). So, we decided to focus on a following behavior to evade these false positives.

### 4.3 Registration and deletion of uninstallation information

An installer usually provides a means to uninstall installed applications. And it is performed by registering uninstall information to certain registry. On the other hand, an uninstaller deletes the registered information from the registry because this information is not necessary any more. These registration and deletion of uninstall information are performed the following registry key.

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
\CurrentVersion\Uninstall\ApplicationName
```

On the other hand, it is rare that malwares perform registration or deletion of the uninstallation information. It is thought that this is because registration of the uninstallation information provides a way to tell their existence to a user. Hence, it is thought that we can distinguish installers and uninstallers from malwares by focusing on this behavior.

### 4.4 Malware detection mechanism

In our proposal, we detect malwares in the step such as figure1. In the following, we describe these steps.

#### – Step1 : Creation of files to particular folders

We observe the event of creating files by a program, and inspect the folders which the files are created to by the program. If these folders are not included in the defined particular folders, we judge this program is not a malware, and is a normal program. Otherwise, we next check Step2.

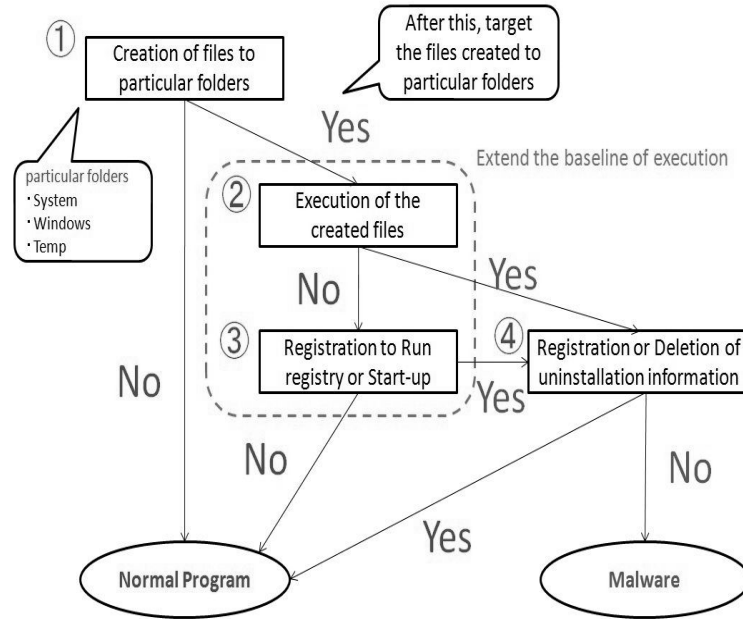


Fig. 1. Malware detection mechanism

– **Step2 : Execution of the created files**

If the files which are created to the particular folders are executed immediately, we assume that the program which created the files is probably malicious. At this point, because the program still has the possibility that it is an installer and uninstaller, we check Step 4 to evade false positives. If not, we extend the baseline of execution and check Step3.

– **Step3 : Registration to Run registry or Start-up**

If the files which are created to particular folders are registered to Run registry or Start-up, considering the extended baseline of execution, we assume that this action means the execution of a program, and check Step4. Otherwise, we judge the program is not a malware, and is a normal program.

– **Step4 : Registration and deletion of uninstallation information**

We examine whether the program registers or deletes uninstallation information. If the program registers uninstallation information, we consider it as an installer, and if it deletes uninstallation information, we consider it as an uninstaller. That is, we judge the program is a normal program. Otherwise, we finally judge the program is a malware.

## 5 Experiment

In our evaluation experiment, we need to analyze the behavior of programs dynamically. In this time, we decided to use ProcessMonitor[9] as a monitoring tool. Using the ProcessMonitor, we can observe all processes' events such as file and registry access, creation of process in real time. From the log file created by the ProcessMonitor, we analyzed the behavior of programs. In the following, we describe the result of our evaluation experiment.

### 5.1 Dataset

In our evaluation experiment, we selected Virus, Worm, Trojan horse, and the others (such as Bot), and its total number of malware is 83. The number of Virus is 19, Worm is 23, Trojan horse is 25, and the others are 16. We put the result of Kaspersky's virus scan for those malwares on the appendix of the end.

On the other hand, for normal programs, we selected eighteen kinds of installer, six kinds of uninstaller, and seventeen kinds of the others such as web browser and ftp client. The total number of normal programs is forty one. In our proposal, it is thought that the false positives of installers and uninstallers become a problem. So, we selected the installer and uninstaller mainly. And, the installers contain five kinds of resident type installer. The resident type installer sets up system so that it is executed automatically at the time of OS start.

### 5.2 Creation of files to particular folders

We show the result regarding the creation of files to particular folders by malwares and normal programs in following Table 1.

**Table 1.** Detailed result regarding creation of files to particular folders

|                              |       | Number of samples which create files to particular folders |         |      |
|------------------------------|-------|--|---------|------|
| Category                     | Total | System   | Windows | Temp |
| Malware                      | 83    | 36   | 15      | 18   |
| Normal program               | 41    | 2  | 1       | 30   |
| Installer(not resident type) | 13    | 1  | 0       | 13   |
| Installer(resident type)     | 5     | 1  | 1       | 4    |
| Uninstaller                  | 6     | 0  | 0       | 6    |
| Others                       | 17    | 0  | 0       | 7    |

In the Table 1, for example, it means that there are thirty six malwares which create at least one file to System folder. Table 1 shows that the feature of creating files to particular folders by malware is more discriminative than normal program. Especially, normal programs hardly create a file to System and Windows folder.

### 5.3 Execution of the created files

We examined whether malwares and normal programs executed the file created to particular folders immediately. We show the result in following Table 2.

**Table 2.** Execution of the created files to particular folders

| Category                      | Number | Creation of files to particular folders | Execution of the created file | Percentage |
|-------------------------------|--------|---|-------------------------------|------------|
| Trojan horse                  | 25     | 19                                      | 13                            | 52%        |
| Worm                          | 23     | 21                                      | 11                            | 48%        |
| Virus                         | 19     | 11                                      | 7                             | 37%        |
| Others                        | 16     | 13                                      | 11                            | 69%        |
| Total                         | 83     | 65                                      | 42                            | 51%        |
| Installer (not resident type) | 13     | 13                                      | 8                             | 62%        |
| Installer (resident type)     | 5      | 4                                       | 1                             | 20%        |
| Uninstaller                   | 6      | 6                                       | 4                             | 67%        |
| Others                        | 17     | 7                                       | <b>0</b>                      | <b>0%</b>  |
| Total                         | 41     | 31                                      | 13                            | 32%        |

Next, In Table 3, we show the result of the case that we consider not “the execution immediately”, but also “the execution at the time of OS start”.

**Table 3.** the result considering the execution at the time of OS

| Category                      | Number | Execution baseline    |   | Percentage |
|-------------------------------|--------|-----------------------|---|------------|
|                               |        | Executing immediately | Executing until the next time of OS start |            |
| Trojan horse                  | 25     | 13                    | 14  | 56%        |
| Worm                          | 23     | 11                    | 17  | 74%        |
| Virus                         | 19     | 7                     | 8   | 42%        |
| Others                        | 16     | 11                    | 12  | 75%        |
| Total                         | 83     | <b>42</b>             | <b>51</b>                                 | <b>61%</b> |
| Installer (not resident type) | 13     | 8                     | 8   | 62%        |
| Installer (resident type)     | 5      | <b>1</b>              | <b>1</b>                                  | 20%        |
| Uninstaller                   | 6      | 4                     | 4   | 67%        |
| Others                        | 17     | 0                     | 0   | 0%         |
| Total                         | 41     | 13                    | 13  | 32%        |

Table 3 shows that the execution rate of files created to particular folders is improved by considering also “the execution at the time of OS start”. That is, the true positive of malwares is improved. Also, the resident type installer usually sets up system so that it is executed automatically at the time of OS start. But,

the program registered to Run registry or Start-up was not the file created to particular folders. It is because installers tend to create files to “Program Files”, not particular folders. Therefore, there was no change in the possibility of false positive of normal programs even if we extended the baseline of execution.

#### 5.4 Registration and deletion of uninstallation information

We examined whether malwares and normal programs registered or deleted uninstallation information. We show the result in following Table 4.

**Table 4.** Registration and deletion of uninstallation information

| Category                      | Number | Register | Delete | Percentage |
|-------------------------------|--------|----------|--------|------------|
| Trojan horse                  | 25     | 1        | 0      | 4%         |
| Worm                          | 23     | 0        | 0      | 0%         |
| Virus                         | 19     | 0        | 0      | 0%         |
| Others                        | 16     | 0        | 0      | 0%         |
| Total                         | 83     | 1        | 0      | 1.2%       |
| Installer (not resident type) | 13     | 12       | 0      | 92%        |
| Installer (resident type)     | 5      | 3        | 0      | 60%        |
| Uninstaller                   | 6      | 0        | 6      | 100%       |
| Others                        | 17     | 0        | 0      | 0%         |
| Total                         | 41     | 12       | 6      | 50%        |

Table 4 shows that it is rare that malwares register or delete uninstallation information. Although there was an only one sample which registered uninstallation information, we think that it does not much influence the true positive rate of malwares because this sample is about 1.2% as a whole in malwares.

On the other hand, in normal programs, much installers and uninstallers registered or deleted the uninstallation information. Although there were a few installers which didn’t register the information, they didn’t cause false positives. It is because that the intended files for execution at step2 and step3 in Figure 1 were not created to particular folders.

#### 5.5 True positives and false positives

From the above results, we examined the true positive of malwares and false positive of normal programs in our proposal. We show that result in following Table 5.

Table 5 shows that the true positive of malwares was about 60% in our evaluation experiment. Also, in the experiment, there were not false positive of normal programs. However, the number of normal programs was not so much in the experiment. So, there is the possibility that the false positive is caused if we increase the number of normal program, especially installers.

**Table 5.** Summary of true positive and false positive rate

| Category                      | Number | True positive |            | False positive |            |
|-------------------------------|--------|---------------|------------|----------------|------------|
|                               |        | number        | percentage | number         | percentage |
| Trojan horse                  | 25     | 14            | 56%        | -              | -          |
| Worm                          | 23     | 17            | 74%        | -              | -          |
| Virus                         | 19     | 8             | 42%        | -              | -          |
| Others                        | 16     | 12            | 75%        | -              | -          |
| Total                         | 83     | <b>51</b>     | <b>61%</b> | -              | -          |
| Installer (not resident type) | 13     | -             | -          | 0              | 0%         |
| Installer (resident type)     | 5      | -             | -          | 0              | 0%         |
| Uninstaller                   | 6      | -             | -          | 0              | 0%         |
| Others                        | 17     | -             | -          | 0              | 0%         |
| Total                         | 41     | -             | -          | <b>0</b>       | <b>0%</b>  |

## 5.6 Evaluation and Discussion

Our proposal is behavior based method using the dynamic analysis. Therefore, unlike the pattern matching method, our method can detect unknown, self-encrypting, and self-obfuscating malwares. On the other hand, the true positive rate of malwares in our proposal is about 60%, and it is not necessarily high. Especially, the true positive rate of Virus is lowest with 42%. This is because Virus tends to overwrite the existing files rather than creating new files for its infection activity. Except for Virus, the true positive rate in our proposal can be improved up to 67%. Therefore, we think that our proposal is effective for the malwares except Virus.

## 6 Implementation of proposal

In our evaluation experiment this time, we use ProcessMonitor to analyze the behavior of programs. Because the amount of the log file that ProcessMonitor outputs is huge, it takes time for the analysis. And, we need to take a log manually. Then, we made a program which gets a log file automatically from ProcessMonitor when an EXE file is specified as an input file. This program, next, judges whether the specified file is malware or not based on our proposal.

In our program, first, an EXE file is specified as an input file which we want to judge whether it is malware or not. When the file is specified, our program activates ProcessMonitor, and continuously executes the specified file. Then, it starts to observe the behavior of the specified file. When a certain amount of time (we assumed 30 seconds this time) passes, our program terminates ProcessMonitor and obtains the log file of behavior of the specified file from ProcessMonitor. Next, our program analyzes the behavior of the specified file from the obtained log file. This analysis is based on our proposal in Figure 1. When the analysis is finished, our program outputs the judgment about whether the specified file is malware or not. And, if our program judges “Yes” in each step of Figure 1, it

also outputs the detailed result. We show the example of analysis result by our program in following Figure 2.

```

[PID:2816] WriteFile
  Creation of file to particular folders:C:\Windows\System32\loadcfg32.exe
[PID:2816] Process Create
  Execution of the created file : C:\Windows\System32\loadcfg32.exe
[PID:3116] RegSetValue
  Registration to Run registry:loadcfg32.exe
=====
[Result]
Malware
=====

```

**Fig. 2.** Example of analysis result (the case of Backdoor.Win32.SdBot.b)

Figure 2 is an analysis result of the malware called Backdoor.Win32.SdBot.b by our program. As a result, we can see that the specified file was finally judged a malware. Moreover, we can see that our program’s analytic process proceeded to step2, and next step4, and final judgment (Malware).

Using our program, we can effectively analyze the behavior of malwares and judge whether the specified file is malware or not. This is because our program can automatically analyze the behavior of malwares only by our specifying the EXE file. Actually, using our program, we can reduce the time that it takes about 5-10 minutes for our getting a log file manually by ProcessMonitor and analyzing it, to about 2-3 minutes. Moreover, we only to specify EXE files, so, when we analyze a lot of malwares, we can easily obtain all the analysis results of these malwares.

## 7 Conclusion and Future works

In this paper, we proposed a malware detection scheme based on evaluation of suspicious process behaviors on Windows operating system. In our proposal, as malicious behaviors of malwares, we focused on the behaviors of creating files to System, Windows, and Temp folders and of executing the file created to these folders. And, to decrease the false positives, we also focused on the behavior of registering and deleting uninstallation information that malwares would usually not do. The result of our experiment using our dataset shows that our proposal detected about 60% of malwares and there were no false positives. Moreover, we implemented our proposal by using a log file of ProcessMonitor. Using our tool, we can verify whether a program is malware or not automatically, and reduce the time of analyzing the behavior of programs.

In future works, we plan to increase the number of malwares and normal programs in our experiment. In this time, we analyzed manually the behavior of malwares and normal programs, so the number of them that we analyzed was limited to some extent. However, we implemented our proposal, so we will

evaluate the capability of our proposal more appropriately by using our tool. Moreover, the true positive rate of our proposal was about 60%, and it is not necessarily high. Therefore, to improve the capability of our proposal, we need to focus on other malicious behaviors.

## References

1. Anubis: Analyzing Unknown Binaries,  
<http://anubis.iseclab.org/>
2. Bayer, U., Moser, A., Kruegel, C., Kirda, E. "Dynamic analysis of malicious code", *Journal in Computer Virology, Volume 2, Number 1*, pp.67-77, August 2006
3. Christodorescu, M., Jha, S., Kruegel, C., "Mining Specifications of Malicious Behavior", *ESEC/FSE '07*, pp.5-14, 2007
4. CWSandbox - Automated Malware Analysis,  
<http://www.cwsandbox.org/>
5. Norman SandBox Information Center,  
<http://www.norman.com/microsites/nsic/>
6. Rieck, K., Holz, T., Willems, C., Dussel, P., Laskov, P., "Learning and Classification of Malware Behavior", *Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA2008)*, pp.108-125
7. Wagener, W., State, R., Dulaunoy, A. "Malware behaviour analysis", *Journal in Computer Virology, 2008-Springer*, pp.279-287
8. Warrender, C., Forrest, S., Pearlmuter, B. "Detecting intrusions using system calls: alternative data models", *Proc. IEEE Symposium on Security and Privacy, 1999.*, pp.133-145
9. Windows Sysinternals,  
<http://technet.microsoft.com/ja-jp/sysinternals/default.aspx>
10. XDing, X., Huang, H., Ruan, Y., Shaikh, A., Zhang, X., "Automatic Software Fault Diagnosis by Exploiting Application Signatures", *Proc. 22nd Large Installation System Administration Conference (LISA '08)*, pp.23-39
11. Zhang, B., Yin, J., Hao, J. "Intelligent Detection Computer Viruses Based on Multiple Classifiers", *Ubiquitous Intelligence and Computing Volume 4611/2007*, pp.1181-1190

## Appendix : Malware Dataset (category is based on Kaspersky's virus scan)

| Trojan horse                     | Worm                        | Virus                    | Others                    |
|----------------------------------|-----------------------------|--------------------------|---------------------------|
| Trojan.Win32.Agent.anb           | Email-Worm.Win32.Bagle.a    | Virus.VBS.Small.a        | Backdoor.Win32.Agent.a    |
| Trojan.Win32.Agent.bd            | Email-Worm.Win32.Bagle.ax   | Virus.Win32.Elkern.a     | Backdoor.Win32.Agent.h    |
| Trojan.Win32.Agent.ez            | Email-Worm.Win32.Mimail.b   | Virus.Win32.Eva.a        | Backdoor.Win32.Aimbot.ad  |
| Trojan.Win32.Agent.i             | Email-Worm.Win32.Mydoom.a   | Virus.Win32.Eva.c        | Backdoor.Win32.Allaple.a  |
| Trojan-DDoS.Win32.Agent.c        | Email-Worm.Win32.Mydoom.b   | Virus.Win32.Eva.g        | Backdoor.Win32.Gobot.a    |
| Trojan-Downloader.Win32.Agent.ab | Email-Worm.Win32.Netsky.g   | Virus.Win32.Funlove.4070 | Backdoor.Win32.IRCBot.aks |
| Trojan-Downloader.Win32.Agent.b  | Email-Worm.Win32.Netsky.q   | Virus.Win32.Gpcode.a     | Backdoor.Win32.IRCBot.ca  |
| Trojan-Downloader.Win32.Agent.z  | Email-Worm.Win32.Netsky.x   | Virus.Win32.Parite.b     | Backdoor.Win32.IRCBot.gyh |
| Trojan-Downloader.Win32.Bagle.ac | Email-Worm.Win32.Sober.j    | Virus.Win32.Parite.d     | Backdoor.Win32.IRCBot.v   |
| Trojan-Downloader.Win32.Bagle.y  | Email-Worm.Win32.Sober.t    | Virus.Win32.Sality.c     | Backdoor.Win32.MSBot.a    |
| Trojan-Downloader.Win32.Small.aa | Email-Worm.Win32.Warezov.u  | Virus.Win32.Sality.d     | Backdoor.Win32.Rbot.aeu   |
| Trojan-Downloader.Win32.Tny.y    | IM-Worm.Win32.Bropia.a      | Virus.Win32.Sality.q     | Backdoor.Win32.SdBot.a    |
| Trojan-Downloader.Win32.Zlob.ab  | Net-Worm.Win32.Bozari.a     | Virus.Win32.Silly.e      | Backdoor.Win32.SdBot.b    |
| Trojan-Downloader.Win32.Zlob.zm  | Net-Worm.Win32.Mytob.a      | Virus.Win32.Small.b      | Backdoor.Win32.SdBot.n    |
| Trojan-Dropper.Win32.Agent.aa    | Net-Worm.Win32.Nimda        | Virus.Win32.Small.c      | Backdoor.Win32.VanBot.dt  |
| Trojan-Dropper.Win32.Agent.ac    | Net-Worm.Win32.Padobot.b    | Virus.Win32.VB.b         | Rootkit.Win32.Agent.q     |
| Trojan-Dropper.Win32.Agent.q     | Net-Worm.Win32.Padobot.c    | Virus.Win32.Virut.a      |                           |
| Trojan-Dropper.Win32.Agent.z     | Net-Worm.Win32.Padobot.z    | Virus.Win32.Virut.ai     |                           |
| Trojan-Dropper.Win32.Agent.zy    | Net-Worm.Win32.Protoride.aa | Virus.Win32.Virut.b      |                           |
| Trojan-Dropper.Win32.Agent.zy    | P2P-Worm.Win32.Spybot.a     |                          |                           |
| Trojan-Dropper.Win32.Blastit.a   | P2P-Worm.Win32.Spybot.z     |                          |                           |
| Trojan-Dropper.Win32.Blastit.b   | Worm.Win32.Agent.d          |                          |                           |
| Trojan-Dropper.Win32.Sramler.e   | Worm.Win32.Antinny.k        |                          |                           |
| Trojan-Spy.Win32.Bzub.ak         |                             |                          |                           |
| Trojan-Spy.Win32.KeyLogger.a     |                             |                          |                           |