惡意電子郵件偵測之研究—以自我組織映射圖與 k-medoids 群集模式為例

施東河
雲林科技大學資訊管理研究所

摘要

現今最重要的網際網路安全威脅議題之一，便是透過電子郵件為傳播媒介的惡意電子郵件病毒與網路蠕蟲，這些病毒與蠕蟲每年以數千隻的比率在成長，構成的一連串的安全威脅。現今的防毒軟體大都以找出病毒特徵碼的方式來防範新的電子郵件病毒，但在新的電子郵件病毒特徵碼尚未找出與更新之前，使用者電腦是暴露在電子郵件病毒的威脅之下。本研究擬提出惡意電子郵件偵測模式，結合自我組織映射圖與 k–medoids 群集模式來偵測未知、新的惡意電子郵件病毒。

本研究所提之惡意電子郵件偵測模式係透過分析各種惡意電子郵件病毒的特性，找出正常電子郵件與惡意電子郵件病毒間的行為特徵，以便自動偵測新的、未知的惡意電子郵件病毒。本文採用偵測率與誤判率作為績效指標，將本研究所提之惡意電子郵件偵測模式與貝式分類、防毒軟體做比較，實驗結果顯示，本研究提出的惡意電子郵件偵測模式明顯優於貝式分類與一般防毒軟體。

關鍵字：自我組織映射圖、k–medoids 群集、電子郵件病毒偵測、電子郵件病毒

Detection of New Malicious Emails Based on Self-Organizing Maps and K-Medoids Clustering

Dong-Her Shih
Department of Information Management,
National Yunlin University of Science and Technology

Abstract

A serious security threat today is malicious emails, especially new, unseen Internet worms and virus often arriving as email attachments. These new malicious emails are created at the rate of thousands every year and pose a serious security threat. Current anti-virus systems attempt to detect these new malicious mail viruses with signatures generated by hand but it is costly and oftentimes. In this paper, we present a method of combining self-organizing maps (SOM) and a k-medoids clustering for detecting new, previously unseen malicious emails accurately and automatically. This method automatically found behaviors in data set and used these behaviors to detect a set of new malicious mail viruses included scripts that hadn't been discussed before. Naïve Bayes classification and anti-virus software's results are also shown for comparison. Comparison results show that our proposed method outperformed than other methods.

Keywords: self-organizing maps (SOM); K-medoids; email virus detection; anti-virus

1. Introduction

In recent years, the number of Internet users worldwide has continued to rise dramatically as the Internet expands. Within this growth, serious problems such as unauthorized intrusions, denial of service attacks, and computer viruses have arisen. In particular, a computer virus is able to cause damage to a large number of systems because of its ability to propagate. As a result, the power of such attacks can now have a serious impact on an information society. Analysis of reported virus incidents during the five-year period (Coulthard and Vuori 2002) provides interesting insights for anti-virus research, as it reflects a period of rapid uptake in the application of the Internet and the use of e-mail for business purposes. Not surprisingly, there is a substantial increase in the number of incidents reported during the period, as shown in Figure 1. Overall, significant growth has occurred in the number reported virus incidents. As shown in Figure 1, strong correlation does exist between virus incidents over time, as supported statistically by a high coefficient of correlation, which is the total explained variance of virus incidents over time.

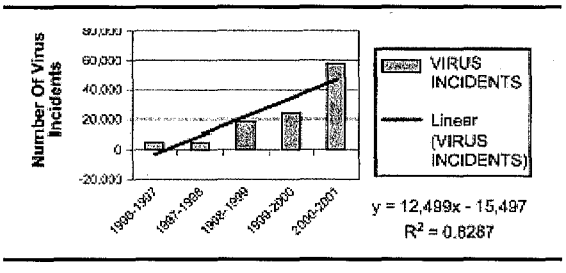

Figure 1 : Total virus incidents

Recently, viruses that spread far more rapidly than before and infected executable files have started to appear. This type of virus propagates by e-mail, one means of information exchange among users. A virus which propagates by e-mail and independently of user operation (file copying and mail sending) after being infected is called "a mail virus" and is distinguished from a conventional virus which propagates as a result of user operation. A mail virus can be assumed to have the following features (Okamoto and Ishida 2002):

- The virus is composed of a script file, and the virus can become active as a result of being launched on a personal computer (PC) on which the script file can be executed.
- The virus propagates to one or more mail addresses stored on the PC by e-mail.
- The virus propagates only when a user first executes it. After the propagation, the virus exists latently.

- When the virus is latent, it will not propagate even if the virus is executed again.
- The virus does not mutate. Mutation includes changes in the code as a result of the virus itself, and modifications by users.

These features differ in the following two ways from a conventional virus, which infects an executable file or program. The first is that a mail virus propagates independently of user operation once the user has executed it. In the past this type of virus proliferated to the addresses which it could reference on the PC, such as mail addresses registered by the user. As a result, the scale over which it could spread after just one infection was incomparably larger than that of a conventional virus.

The second is that discovering such a virus is easier than a conventional virus, which infects an executable file. A mail virus can be contained in some text messages or some file attachments (virus codes). The user can discover the virus even without using anti-virus software if he or she knows that some text messages contain a virus. This information can be readily obtained from the mass media and the Internet. In addition, a user who has discovered a virus even once will likely be very alert to reinfection by viruses because they can be discovered very readily based on some of the text messages or file attachments.

A malicious mail or mail virus is also defined as executables attachment in an email that performs some functions, such as damaging a system, transmitting copies of itself by email to address from their address book and inbox automatically. For example, ILOVEYOU is the case (Garber 1999). Besides, some mail viruses do not require the mail receiver to open the attachment for it to execute. A known vulnerability in Internet Explorer-based email clients (Microsoft Outlook and Microsoft Outlook Express) automatically executes the file attachment.

One of the primary problems faced by the virus community is to devise methods for detecting new virus that have not yet been analyzed. Eight to ten viruses are created every day and most cannot be accurately detected until signatures have been generated for them (White *et. al.* 1999). During this time period, systems protected by signature-based algorithms are vulnerable to attacks.

Current virus scanner technology has been divided into two parts: signature-based detector and heuristic classifier that detect new viruses (Gryaznov 1999). The classic signature-based detection algorithm relies on signatures (unique telltale strings) of known malicious executables to generate detection models. Signature-based methods create a unique tag for each malicious program so that future examples of it can be correctly classified with a small error rate. These methods do not generalize well to detect new malicious mails because they are created to give a false positive rate as close to zero as possible. The standard approach to protecting against malicious mails is to use a virus scanner but most of these virus scanners are signature based. Because of the mail virus has quickly dissemination and polymorphisms characteristic, a traditional signature-based method is not efficient to detect

them (Trend Micro).

Our research presents a framework, a mail virus filter that can detect malicious Windows attachments by integrating with a mail server. Analyzing the characteristic of the mail virus, to picked out differentiate one mail virus from another, and from normal mail. Using heuristic classifiers to detect new unknown mail virus, the classifier is a rule set, or detection model, generated by algorithm like SOM methods that was trained over a given set of training data. Extracting some features from training data that contain malicious and benign mail.

Our goal is to design and build a scanner that accurately detects mail virus before they have been entered into a host. The methods discussed in the paper are acting as a network mail filter. We are going to implement a network-level email filter to catch malicious mail virus before users receive them through their mail.

2. Related work

Detecting malicious executables is not a new problem in security. Protection from unknown viruses is, indisputably, the issue of the day in computer virology. Early methods used signatures to detect malicious programs. Current approaches are matching them to a set of known malicious programs. Experts were typically employed to analyze suspicious programs by hand. Using their expertise, signatures were found that made a malicious executable example different from other malicious executables or benign programs. One example of this type of analysis was performed by Spafford (1988). He analyzed the Internet Worm and provided detailed notes on its spread over the Internet, the unique signatures in the worm's code, the method of the worm's attack, and a comprehensive description of system failure points. Malicious code is usually classified (McGraw and Morrisett 2000) into the following categories:

- Viruses are programs that self-replicate within a host by attaching themselves to programs and/or documents that become carriers of the malicious code;
- Worms self-replicate across a network;
- Trojan horses masquerade as useful programs, but contain malicious code to attack the system or leak data;
- Back doors open the system to external entities by subverting the local security policies to allow remote access and control over a network;
- Spyware is a useful software package that also transmits private user data to an external entity.
- Attack scripts are programs written by experts that exploit security weaknesses, usually across the network, to carry out an attack.
- Java attack applets are programs embedded in Web pages that achieve foothold

through a Web browser.

- Dangerous ActiveX controls are program components that allow a malicious code fragment to control applications or the operating system.

Combining two or more of these malicious code categories can lead to powerful attack tools. For example, a worm can contain a payload that installs a back door to allow remote access. When the worm replicates to a new system (via email or other means), the back door is installed on that system, thus providing an attacker with a quick and easy way to gain access to a large set of hosts. Once the back-door tool gains a large installed base, the attacker can use the compromised hosts to launch a coordinated attack, such as a distributed denial-of-service (DDoS) attack (Michie *et. al.* 1994).

At IBM, Kephart and Arnold (1994) developed a statistical method for automatically extracting malicious executable signatures. Their research was based on speech recognition algorithms and was shown to perform almost as good as a human expert at detecting known malicious executables. Their algorithm was eventually packaged with IBM's anti-virus software. Lo *et al.* (1995) presented a method for filtering malicious code based on "tell-tale signs" for detecting malicious code. Similarly, filters for detecting properties of malicious executables have been proposed for UNIX systems (Kerchen *et. al.* 1990) as well as semiautomatic methods for detecting malicious code (Crawford *et. al.* 1993).

As is well known, the main deficiency of commonly used anti-virus scanners is that these scanners are able to detect and delete only those malicious programs described in their anti-virus databases. The anti-virus community relies heavily on known byte-code signatures to detect malicious programs. The main disadvantage to this approach is that the scanner cannot detect a virus if the database doesn't contain its description and the user is unprotected from this new virus threatening during this time period.

In order to shorten the period of viruses' signature code updated, anti-virus manufacturers have created and implemented certain technology allowing for automation of the analysis and development and testing of updates for anti-virus databases. E-mail and advantages of other contemporary data-transmission technologies (for example, the "Push" technology) allow delivery of ready-made anti-virus database updates to end-users within seconds. But all these efforts still haven't solved the key problem: protection from unknown (new) viruses before they actually appear. To detect unknown malicious emails, there are several advanced approaches described below.

2.1 Heuristic classifier

Since a new malicious program may not contain any known signatures. Therefore, traditional signature-based methods may not detect a new malicious executable. The main feature of computer viruses is their ability to embed in other programs while partially or completely retaining those programs' operability. During the process of infection (when a virus

embeds itself into a program), viruses modify the parent program so that when the system calls this program, the malicious code is executed. In an attempt to solve this problem, the anti-virus industry generates heuristic classifiers by hand (Gryaznov 1999). This classifier reconstructed the behavior of the program it was checking, and used it as a basis for conclusions about the potential danger from this program. Furthermore, to be able to detect different virus types, you must use different approaches and, as a result, different heuristic algorithms. This process can be even more costly than generating signatures, so finding an automatic method to generate classifiers has been the subject of research in the anti-virus community. To solve this problem, different IBM researchers applied Artificial Neural Networks (ANNs) to the problem of detecting boot sector malicious binaries (Tesauro *et. al.* 1996). An ANN is a classifier that models neural networks explored in human cognition. Because of the limitations of the implementation of their classifier, they were unable to analyze anything other than small boot sector viruses that comprise about 5% of all malicious binaries.

Using an ANN classifier with all bytes from the boot sector malicious executables as input, IBM researchers were able to identify 80–85% of unknown boot sector malicious executables successfully with a low false positive rate. They were unable to find a way to apply ANNs to the other 95% of computer malicious binaries. In similar work, Arnold and Tesauro (2000) applied the same techniques to Win32 binaries, but because of limitations of the ANN classifier they were unable to have the comparable accuracy over new Win32 binaries.

2.2 Redundant Scanning

Unlike the heuristic classifier that is currently used in every competent anti-virus program, the redundant scanning technique is not so common (Zenkin 2001). In the beginning of the 1990s, they developed and implemented in anti-virus scanners tools that use extraordinary methods when infecting files. Viruses write the transfer-to-virus instruction somewhere in the file contents and obtain control when the procedure containing a code transferring control to the virus body is executed rather than started. Before writing the transfer-to-virus instruction into a file, the virus must choose a "correct" address in this file. Redundant scanning allows for the scanning of not only the system processing entry points, but the entire contents of any given object. In fact, this method allows an anti-virus program to look far inside a file in order to see its entire contents are inaccessible for others.

2.3 Integrity Checker

An integrity checker is yet another anti-virus technology that allows for unknown virus detection (Luke and Harris 1999). It is based on the fact that viruses can be considered average programs that have the ability for the unauthorized creation of new and modifying or self-planting into existing objects (such as files, boot sectors or system registry). An integrity

checker's operating mode is based on a collection of original "prints" (CRC-values) of files, boot sectors and a system registry. These "prints" are stored in the database. When started up, the integrity checker compares information from its database with current "prints" and informs a user of changes that have occurred, marking virus-like changes and other unsuspicious changes. Some integrity checkers offer improved file analysis for the fastest and the most effective checking of a computer requiring minimum system resources. An integrity checker's approach to infected-object restoration is based on the knowledge of what should be inside a clean file or boot sector rather than on the knowledge of a virus' appearance. Everything that violates this rule is considered as a change and is subject to being reported to the user and having the original content restored. They usually need no more than 500 Kbytes. Integrity checkers are used not only in basic standalone home computers, but also on the server level of computing systems. Integrity checkers can be considered powerful tools not only against known and unknown computer viruses, but also as strong alternative intrusion detection and anti-hacker utilities. However, integrity checkers cannot detect a virus upon entry into the system and can only do so after a period of time. They cannot detect a virus in new files (e-mail, files on disks, files restored from backup, and files extracted from archive), since there is not any information in their databases about these objects. Some viruses even use this feature and infect only newly developed files, thereby staying "invisible" to the integrity checkers.

2.4 Behavior Blocker

It seems that the absolute solution for this problem could be found only after the discovery of an artificial intelligence that would be able to analyze computer data as today's best anti-virus experts do. Behavior blocker is a memory-resident program intercepting various computer operations. The behavior blocker monitors any program's activity and prevents harmful actions that could be done by malicious code. In theory, the behavior blocker may prevent the distribution of any known and/or unknown (written after the blocker was developed) virus, warning the user about the virus before it is able to infect other files or damage his computer. However, the operating system itself or some useful utilities can also perform virus-like actions (such as creating files or modifying the system registry). The user must possess sufficient knowledge and experience, otherwise the operating system or utility will be disabled from performing the action required, or a virus will penetrate the system. This is the main reason why blockers have yet to become popular. The blocker also monitors macros working with external applications, including e-mail programs. It eliminates the possibility of macro-virus distribution through e-mail. Unlike traditional anti-virus technologies, a behavior blocker solves this problem by blocking macro access to e-mail. In case the blocker detects an attempt by a macro-program to access an e-mail program, it can block the function or even terminate the whole macro. The behavior blocker solves the problems of macro virus detection and distribution prevention, but is not designed to delete

macro-viruses and restore the infected files. This is why it should be used in conjunction with an anti-virus scanner that is able to delete viruses. Bloodhound technique (Symatec) and scriptTrap scanning technique (Trend Inc.) are the case. With the development of computer technologies, especially in the field of artificial intelligence, the magnitude, efficiency and simplicity-of use of behavior blockers will rapidly increase.

2.5 Agent-Based Simulation

Another idea of mail virus prevention is through shell simulation. All programs would run like a shell outside the kernel system. For example, virus instruction code emulation (VICE) (Lee *et. al.* 1997) has been applied to a virtual machine that simulates CPU operation. Incorporated with experts' knowledge base, VICE clamed they can detect polymorph and mutation viruses. It includes three detection stages: replication detection, similarity assessment and intension discrimination. By observing the program action in the virtual machine, VICE can prevent all the malicious activity. The drawback of VICE is that it took too many system resources and it tear down the total efficiency. Due to the fast spread character of email, using this technology may be more considered.

Our method is different from the previous researches because we analyzed the entire features of malicious mail instead of only boot-sector viruses or only Win32 binaries. Our technique is similar to data mining techniques that have already been applied to Intrusion Detection Systems by Lee *et. al.* (1999). Their methods were applied to system calls and network data to learn how to detect new intrusions. They reported good detection rates as a result of applying data mining to the problem of intrusion detection system. We applied a similar framework to the problem of detecting new malicious mail viruses but included script that Schultz *et. al.* (2001; 2002) and others didn't provide. We are going to build SOM network with k-medoids clustering (Kaufman and Rousseeuw 1990) which can represent profiles for activities and capture deviation of current activities from profiles.

The SOM is an unsupervised neural network-learning algorithm and forms a mapping from high-dimensional data to two-dimensional space. However, it is difficult to find clustering boundaries from results of the SOM. On the other hand, the k-medoids clustering can partition the data into the clusters under the assumption of the known number of clusters. In order to understanding the results of SOM, we applying the k-medoids clustering to find out the boundaries from results of SOM. We estimated our results for detecting new mails by using SOM and k-medoids clustering after we have trained our proposed network.

To evaluate the performance we were interested in several quantities:

1. True Positives (TP): the number of malicious mails classified as malicious.
2. True Negatives (TN): the number of benign mails classified as benign..
3. False Positives (FP): the number of benign mails classified as malicious.
4. False Negatives (FN): the number of malicious mails classified as benign.

The false positive rate and the detection rate are compute also. The false positive rate is the number of benign mails that are mislabeled as malicious divided by the total number of benign mails. The detection rate is the number of malicious mails that are caught divided by the total number of malicious mails. The overall accuracy of the algorithm is calculated as the number of mails the system classified correctly divided by the total number of mails tested.

3. Clustering methods

The SOM, K-medoids and our proposed method are introduced in this section.

3.1. Self-Organizing Maps

A Self-Organizing Map (Kohonen 1990, 1995), or SOM, is a neural clustering technique. Having several units compete for the current object performs the SOM clustering. The unit whose weight vector is closest to the current object becomes the winning unit. The weight of the winning unit is adjusted as well as those of its neighbors. SOMs assume that there is some topology among the input objects and the unit will take on this structure in space. The organization of these units is said to form a feature map. It is more sophisticated than k-medoids in terms of presentation; it not only clusters the data points into groups, but also presents the relationship between the clusters in a two-dimensional space. SOM is also capable of presenting the data points in one- or three-dimensional space. However, two-dimensional space is most commonly used due to the trade-off between information content and ease of visualization. The SOM algorithm is outlined in Figure 2

Given node N, the *SOM* algorithm is implemented in 4 steps

Step 1 (Selecting the training vector x_m)

A two-dimensional vector with random components distributed uniformly from 0 to 1 is created, and the training vectors x_m are selected.

Step 2 (Determining the winning vector)

The Euclidean distance $\| x_m - w_i \|$ is calculated for the position vector w_i in all nodes, and the winning vector is determined.

$$j^* = \arg\min_j \left\| \underline{x}(n) - \underline{w}_j \right\|, j = 1,2,\Lambda\Lambda\ ,N$$

Step 3 (Updating the position vector w_j by tuning learning rate η and neighborhood Λ)

$$\underline{w}_j(n+1) = \begin{cases} \underline{w}_j(n) + \eta(n)[\underline{x}(n) - \underline{w}_j(n)], & j \in \Lambda_{j^*}(n) \\ \underline{w}_j(n) & j \notin \Lambda_{j^*}(n) \end{cases}$$

Step 4 (Go to step 2 until the weight have stabilized.)

Figure 2 : The SOM algorithm

3.2 Classical Partition Method: K-Medoids

In order to find out the boundaries from results of SOM, we applied partitioning method. The most famous and commonly used partitioning methods are k-means and k-medoids, and their variation. The k-means algorithm is sensitive to outliers since an object with an extremely large value may substantially distort the distribution of data. Instead of taking the mean value of the objects in a cluster as a reference point, the medoid (representative object) can be used, which is the most centrally located object in a cluster. Thus, the partitioning method can still be performed based on the principle of minimizing the sum of the dissimilarities (distances) between each object and its corresponding reference point. This forms the basis of the k-medoids method. For example, *PAM* (partitioning around Medoids) (Kaufman and Rousseeuw 1990), built in Splus, starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering. *PAM*, use real object to represent the cluster, works effectively for small data sets, but does not scale well for large data sets. The k-medoids algorithm is shown in Figure 3.

The k-medoids algorithm is more robust than k-means in the presence of outlier and noise because a medoid is less influenced by outliers or other extreme values than a mean.

Given k, the *k-medoids* algorithm is implemented in 4 steps

1. Select $\boldsymbol{k}$ representative objects arbitrarily
2. For each pair of non-selected object $\boldsymbol{h}$ and selected object $\boldsymbol{i}$, calculate the total swapping cost $\boldsymbol{S_{ih}}$ (i.e. total distance changed)
3. For each pair of $\boldsymbol{i}$ and $\boldsymbol{h}$,

   If $\boldsymbol{S_{ih}} < 0$, $\boldsymbol{i}$ is replaced by $\boldsymbol{h}$

   Then assign each non-selected object to the most similar representative object
4. Repeat steps 2-3 until there is no change

Figure 3 : The k-medoids algorithm

3.3 Proposed method

The SOM can map the input vectors without any information about the number of cluster, but the clustering boundaries are not clear in results of the SOM. In order to find the clustering boundaries from the results of SOM, we apply the k-medoids clustering to results of SOM. The procedure of the presented method is the training of the SOM and then applying the k-medoids clustering. The input vectors of the SOM are the benign mail data and the input vectors of the k-medoids clustering are the prototype vectors of the SOM. The outline of the proposed method is shown in Figure 4

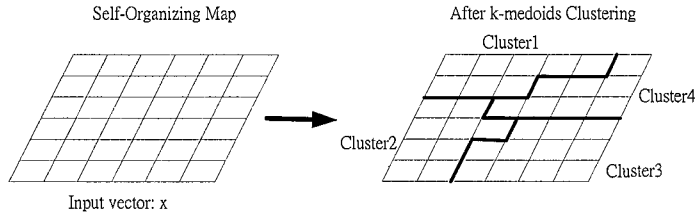

Figure 4：the outline of the proposed method

4. Experiment

The problem of our work was to explore a standard technique to compute accurate detectors for new (unseen) malicious mails. We gathered a large set of mails from public sources and mail server, and separated the problem into two classes: *malicious* and *benign* mails. Every example in our data set is a standard mail format mails, although the framework we present is applicable to other formats. There were no duplicate mails in the data set and every mail in the set is labeled either malicious or benign by the commercial virus scanner. The gathered malicious mail viruses, we found in our data set, were shown in Table1, which consisted of IW (Internet worm), Trojans, Macro, Script and File-infector etc. The contrasting benign data are also collected in order to perform relevance analysis.

Table 1：Samples of mail viruses (Trend Inc.)

Macro	W97M_MELISSA.A，W97M_GORUM.A
Executable File	ZAUSHKA.A-O，JERM.A，COBBES.A，MAGISTR.B，KAMIL.B，YOUGDOS.A
Trojan	PTWEAK.A，GIFT.B，HYBRIS.C，SIRCAM.A，TROODON.A，XTC.A，FEVER.A
Script	**JavaScript**：GERMINAL.A，EXITW.A，SEEKER.A6，ACTPA.A，EXCEPTION.GEN **VBScript**：REPAH.A，HARD.A，NEWLOVE.A，INFO.A，LIFELESS.A，NOONER.A，KALAMAR.A，CHICK.C，CHICK.B，CHICK.E，LOVELETTER，CHU.A，EDNAV.B，GOOFFY.A，HAPTIME.B，HEATH.A，HORTY.A，VIERKA.B，ARIC.A，GORUM.B，ZIRKO.A
Worm	NIMDA.A-O，ALIZ，PETIK.C，PET.TICK.Q，BADTRANS.B，GIZER.A，ENVIAR.B，GOKAR.A，RADIX.A，UPDATR.A，KLEZ.E，KLEZ.H，LASTWORD.A，LOHACK.A，MERKUR.A，MYLIFE.E，PLAGE.A，PROLIN.A，SOLVINA.B，SHOHO.GEN，DESOR.A，PET.TICK.Q，PETIK.E，ZHANGPO.A，ZOHER.A，SOBIG.A，YAHA.E，BUGBEAR.A，BLEBLA.C，GAGGLE.C

We automatically extracted a profile from each mail in our dataset first, and from the profile's *features* to use with classifiers. Using different features, we trained a set of classifiers to distinguish between benign and malicious mails. Noted that the features extracted were static properties of the mail and did not require executed. We are going to build SOM network with k-medoids clustering that can represent profiles for normal activities and capture deviation of current activities from profiles. Naïve Bayes classification method is also proposed for comparison.

4.1 Data set

Our data set, collected until May2003, consisted of malicious and *benign* clean mails in our UNIX mail server. To standardize our data set, we used an updated Norton's virus scanner and labeled our mails as either malicious or benign mails. Assume that the malicious mail virus can be separated by its static characteristics.

The sampling size of our email has been estimated by the population ratio (Mendenhall and Beaver 1994):

$$n = \frac{z_{\alpha/2}^2 P(1-P)}{e^2} \quad \ldots \quad (1)$$

Where n is the sampling size, P is the estimator, e is the sampling error, α is the confidence coefficient and z is the standard normal probability distribution.

Since there is no further study in the estimation of a mail will be the mail virus' probability (P will never be known exactly). Therefore, we use nonconstant number sampling to obtain the email virus' proportion of population in ten time's trial, which is shown in Table 2. From Table 2 we can obtain an email virus' estimator (population proportion expected value)

$$P = \sum_{i=1}^{10} C_i * P_i = 0.0518$$

Substitute P value, 0.95 confidence levels and 0.02 sampling error into equation (1), the sampling mail's size n would be 473. There are 599 user accounts in our mail server. By sampling mails in different group, deleting the same content mails and testing with anti-virus software Norton 2003, we obtained 444 mails (with 84 malicious and 360 benign) in our data set. While if the estimate error is 0.03, the sampling mails would be 210.

Table 2：Ten times nonconstant sampling

i	Mail no.	Virus mail no.	Virus mail ratio(*Pi*)	Mail weight (*Ci*)
1	43	1	0.0232	0.0428
2	51	18	0.3529	0.0508
3	78	3	0.0384	0.0777
4	82	10	0.1219	0.0817
5	58	4	0.0689	0.0578
6	67	2	0.0298	0.0668
7	161	3	0.0186	0.1605
8	36	2	0.0555	0.0358
9	143	3	0.0209	0.1425
10	284	6	0.0211	0.2831

After verification of the data set the next step of our method was to extract features from the mails. We statically extracted different features that represented different information contained within each mail. Then, the algorithms to generate detection models used these static features. The problem of predicting a mail's behavior can be reduced to the halting problem and hence is undesirable. Perfectly predicting a mail's behavior is unattainable but estimating what a mail can or cannot do is possible.

4.2 Attribute Selection Measure

The information gain measure (Han and Kamber 2001) is used to select the test attributes. Such a measure is referred to an attribute selection measure or a measure of the goodness of split. The attribute with the highest information gain (or greatest entropy reduction) is chosen as the test attribute. This attribute minimizes the information needed to classify the samples in the resulting partitions and reflects the least randomness or "impurity" in these partitions. Such an information-theoretic approach minimizes the expected number of tests needed to classify an object.

Let S be a set consisting of s data samples. Suppose the class label attribute has m distinct values defining m distinct classes, D_i (for $i=1,...,m$). Let s_i be the number of samples of S in class D_i. The expected information needed to classify a given sample is

$$I(s_1,s_2,...s_m) = -\sum_{i=1}^{m} p_i \log_2(p_i) \quad \text{.....(2)}$$

Where p_i is the probability that an arbitrary sample belongs to class D_i and is estimated by s_i/s. Note that a long function to the base 2 is used since the information is encoded in bits. Let

attribute A have v distinct values, $\{a_1, a_2, ..., a_v\}$. Attribute A can be used to partition S into v subsets, $\{S_1, S_2, ..., S_v\}$, where S_j contains those samples in S that have value a_j of A. Let s_{ij} be the number of samples of class D_i in a subset S_j. The entropy, or expected information based on the partitioning into subsets by A, is given by

$$E(A) = \sum_{j=1}^{v} \frac{s_{1j} + ... + s_{mj}}{s} I(s_{1j}, ..., s_{mj}) \quad \text{.....(3)}$$

The term $\frac{s_{1j} + ... + s_{mj}}{s}$ acts as the weight of the jth subset and is the number of samples in the subset (i.e., having value a_j of A) divided by the total number of samples in S. The smaller the entropy value is, the greater the purity of the subset partitions. Note that for a given subset S_j,

$$I(s_{1j}, s_{2j}, ...s_{mj}) = -\sum_{i=1}^{m} p_{ij} \log_2(p_{ij}) \quad \text{.....(4)}$$

Where $p_{ij} = \frac{s_{ij}}{|s_j|}$ and is the probability that a sample in S_j belongs to class D_i.

The encoding information that would be gained by attribute A is

$$Gain(A) = I(s_1, s_2, ...s_m) - E(A) \quad \text{.....(5)}$$

In other words, *Gain(A)* is the expected reduction in entropy caused by knowing the value of attribute A. The algorithm computes the information gain of each attribute. The attribute with the highest information gain is chosen as the test attribute for the given set S.

Mail format descriptions involve many attributes and analytical characterization was performed. This procedure first removes irrelevant or weakly relevant attributes prior to performing generalization. Conservative attribute generalization thresholds were used to remove improper attributes. Then, the attributes in the candidate relation are evaluated using the selective relevance analysis measure, such as information gain in equation (5). We used an attribute relevance threshold of 0.15 to identify weakly relevant attributes. The information gain of the attributes, which are below the threshold, are considered weakly relevant and thus removed. The contrasting class is also removed, resulting in the initial malicious class working relation. Finally, from the mail format, we extracted a set of features to compose a feature vector for each mail as shown in Table 3. Note that X_2 and X_9 are removed in our experiment due to low relevance threshold.

Table 3：Extracted feature from mail format

	Feature	Content
X_1	Mail content type	Text/plain，text/html，other
X_2	Mail size	Total size of mail
X_3	MIME Format	Yes，No
X_4	Attachment	Yes，No
X_5	Attachment file no	Number of attachments
X_6	Attachment size	Total size of attachments
X_7	Attachment file type	exe，doc，scr，pif.....
X_8	Script language	VBScript，JavaScript.....
X_9	Subject	Re，Fw，Fwd，...
X_{10}	Carbon Copy	CC，BCC，Undisclosed-Recipient，...
X_{11}	Recipient	Single- Recipient，Multi- Recipient

To extract features from data set, we wrote a *feature extraction* program. The *feature extraction* program extracts features from Microsoft Outlook mail file. All of the information about the mail was obtained from the mail format. In addition, the information was obtained without executing the unknown mail but by examining the static properties of the mail. Through testing we found that there were similar behaviors in malicious mails that distinguished them from clean mails, and similar behaviors in benign mails that distinguished them from malicious mails.

4.3 Anomaly detection with SOM and k-medoids

In order to detect unseen new mail virus, we are going to incorporating the SOM network in anomaly detection first. Anomaly detection often defined as: finds out behavior in normal activities and detects intrusion in malicious activities. We build the network structure from training data and use the testing data to measure its performance. The training data and the testing data contain audit events. The training data consists of audit events during normal activities in the mail server. The testing data contains audit events arising from both normal and malicious activities. During training, the structure of SOM network is constructed based on benign mail data. During testing, we compute the accuracy of SOM network on the evidence of audit events of the testing data. We have used a sample of audit data contains normal activities and malicious activities. We use the first half of normal activities for building normal profiles. The second half of audit events (normal events) and audit events arising from malicious activities are used during testing.

First of all, we use Kohonen's Self-Organizing Map to organize benign mail behavior into a two-dimensional map, according to mails' extracted features. Our input vectors consist of a set of benign mails' features. The desired output is a two-dimensional map of N nodes (in this case, a 10×10 map of 100 nodes). The SOM algorithm has two parameters that change through iterations: the variance of the neighborhood function $\Lambda(n)$ (radial symmetric Gaussian function) and the learning rate $\eta(n)$ for $n=1, 2$ .... The adaptation laws for these parameters are presented as:

$$\eta(n)=0.9(1-n/1000), \quad \Lambda(n)=\Lambda(n-1)(1-0.01n)$$

These parameters adaptation laws lead to a fast convergence of the algorithm without the lost of quality of its output. Using these laws together with the selective update of neurons weighs, there is a reduction of the algorithm complexity through iterations. The selective update consists of a threshold for the neighborhood function that allows only neurons above the threshold to be updated. Since the $\Lambda(n)$ decreases fast with time, so does the number of neurons to be updated.

Figure 5 shows a typical SOM produced by our algorithm with 180 benign mails and the testing data were labeled. The map contains 264 mails' behaviors. The blank nodes contained no mails' mapping, while those that are labeled with "M" and "B" nodes contained malicious and benign mails' mapping within each node. And "+" in the hexagon means the unspecified node which contained both malicious and benign mails' mapping in the same node. The distance between nodes on the map indicates the similarity of the mails' behavior, measured according to the features. Similarity here is measured not by the similarity of the content, but by the similarity of behaviors. It is also difficult to quantitatively measure the effectiveness of the SOM. But, by proper choosing the maximum distance of benign training data and its winning node, the malicious emails can be detected. Assume that the distance between malicious data and its winning node is larger than benign. We can calculate the detection rate of audit mails in testing data. Figure 6 shows the results in ROC curve of SOM1 (with 180 benign mails) and SOM2 (with 360 benign mails). From Figure 6, we can observe that higher detection rare will incorporated a higher false positive rate in SOM. More training data of benign (SOM2) achieve a better accuracy than fewer one (SOM1) in our testing set. If we can obtain more training data, we may obtain a better result.

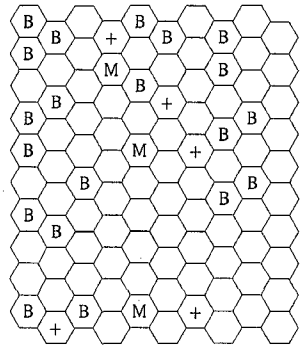

Figure 5：Result of the SOM

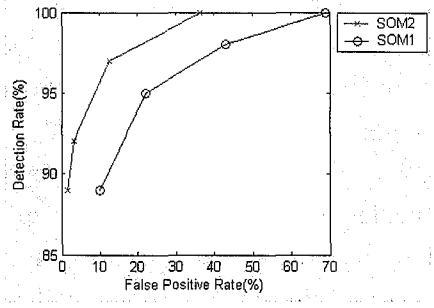

Figure 6：The ROC curves of the SOMs

It is difficult to find clustering boundaries clearly in SOM. Therefore, we apply the k-medoids clustering after the training of the SOM1. Assuming that there are only two clusters in the trained SOM map (malicious or benign). Applying the algorithm (in Figure 3), Figure 7 shows the result of k-medoids clustering for the input vectors in SOM with testing data labeled clearly with "M" and "B". There are no "+" hexagon in Figure 7, since every node in the map is clearly identified with malicious or benign by using k-medoids clustering. Then, the accuracy of SOM with k-medoids clustering can be calculated and shown in Table 4 incorporated with other methods that will be shown and discussed in the next section. With a little false positive rate, our proposed method shows a high detection rate in the testing data.

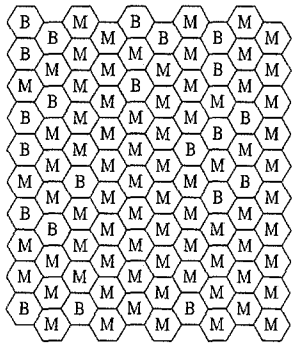

Figure7：Result of SOM (with k-medoids)

Table 4：Comparison results (%)

Profile Type	True Positives	True Negatives	False Positives	False Negatives	Detection Rate (%)	F P Rate (%)	Overall Accuracy(%)
SOM(k-medoids)	82	176	4	2	97.62	2.22	97.72
Naïve Bayes	69	180	0	15	82.14	0	96.62
Pcc2002	35	180	0	49	41.66	0	88.96
Pcc2003	73	180	0	11	86.90	0	97.52
Norton2002	26	180	0	58	30.95	0	86.93
Norton2003	69	180	0	15	82.14	0	96.62

4.4 Naïve Bayes classification

In order to identify the accuracy of other method, we use the well-known Naïve Bayes classification to compare with our proposed method. A Naïve Bayes (Michie *et. al.* 1994; Han and Kamber 2001) classifier computes the likelihood that a mail is malicious given the features that are contained in the format mail. The model, output by the Naïve Bayes algorithm, labels examples based on the features that they contain. For instance, if a mail contained a significant number of malicious features, then the mail can be labeled as a mail of virus. Specifically, we wanted to compute the class of a mail given that the mail contains a set of features *F*. We define *C* to be a random variable over the set of classes: benign, and malicious. That is, we want to compute *P(C/F)*, the probability that a mail is in a certain class given the mail contains the set of features F. We apply Bayes rule and express the probability as:

$$P(C/F)=\frac{p(F/C)*P(C)}{P(F)} \quad \text{.....(6)}$$

To use the Naïve Bayes rule we assume that the features occur independently from one another. If the features of a mail F include the features F_1, F_2, F_3 ...F_n, then equation (6) becomes:

$$P(C/F)=\frac{\prod_{i=j}^{n} P(F_i/C)*P(C)}{\prod_{j=1}^{n} P(F_j)} \quad \text{.....(7)}$$

Each $P(F_i/C)$ is the frequency that features F_i occurs in a mail of class C. $P(C)$ is the proportion of the class C in the entire set of mails. The output of the classifier is the highest probability class for a given set of strings. Since the denominator of equation (6) is the same for all classes we take the maximum class over all classes e of the probability of each class computed in equation (7) to get

$$\text{Most Likely Class} = \max_{c}\left(P(C)\prod_{i=1}^{n} P(F_i/C)\right) \quad \text{.....(8)}$$

Where, we use $\max_C$ to denote the function that returns the class with the highest probability. Most Likely Class is the class in C with the highest probability and hence the most likely classification of the example with features F. Then we applied equation (8) to compute the most likely class for the mail.

The Naïve Bayes algorithm computed the probability that a given feature was malicious and the probability that a feature was benign by computing statistics on the set of training data. Then to predict whether a mail was malicious or benign, those probabilities were computed in the classifier and the Naïve Bayes independence assumption was used. The independence assumption was applied in order to efficiently compute the probability that a mail was malicious or benign. Some of the posteriori probability table was shown in Table 5, where X_i's is denoted in Table 3.

We estimated our results for detecting new mails by using 4-fold cross validation. We averaged the results of these four tests to obtain a measure of how the algorithm performs in detecting new malicious mails. The famous anti-virus software was also used for comparison and the averaging results of all experiments are presented in Table 4. Anti-virus software (Pc-cillin or Norton) is not updated for comparison. From Table 4, we can observe that the Naïve Bayes algorithm performed like new version of anti-virus software (Pcc2003 or Norton2003). It means that by using our methodology, other classification method may also be used. But our proposed method (SOM with k-medoids) is better than others.

Table 5 : The posteriori probability of malicious (M) and benign (B) mail

	X_1	X_2	X_3	X_4	...	X_8	X_9	X_{10}	X_{11}
$P(X_i \mid M)$	0.81	0.06	0.131	0.929	...	0.095	0.202	0.024	0.988
$P(\overline{X_i} \mid M)$	0.19	0.94	0.869	0.071	...	0.905	0.798	0.976	0.112
$P(X_i \mid B)$	0.83	0.22	0.003	0.35	...	0.014	0.292	0.35	0.269
$P(\overline{X_i} \mid B)$	0.17	0.78	0.997	0.65	...	0.986	0.708	0.65	0.731

5. Discussion and further study

One drawback of SOM clustering in our case is the bordline between malicious and benign mails that SOM clustering can't decide. With k-medoids approach, each cell in figure 5 must clearly identify with malicious (M) or benign (B) as in figure 7. If there are too many cell marked with "+" neuron, it is a bad idea in implementing SOM clustering. Although there is no strongly proof that k-medoids is a good way to redefine the bordline boundary of SOM's result. But, through the experimental study, it seems a good idea in identify malicious emails. Since, execution of malicious email may crash our system or unrecoverable. Therefore, extract the mail's static feature before its execution seems the only possible way to prevent infected. Malicious email may spread for several months. If we could accumulate and identify different email viruses longer, we may obtain a better picture of email virus. Further, by close examine in Table 4. We can have some conclusion:

1. A mail virus may spread year by year: Although our data set was collected in 2003, there is some mail viruses can be detected by Pcc2002 (35 mails) or Norton2002 (26 mails) in our data set. Therefore, there is no time limit in the mail virus distribution.
2. It is time concerning to update anti-virus software: There are still some mail viruses that cannot be detected by Pcc2003 (11 mails) and Norton2003 (15 mails) in our data set. It needs update our anti-virus software often to protect our computers.
3. Naïve Bayes classification is another choice of detection tool in compare with anti-virus software: with no false positive rate, Naïve Bayes classification performed like newly updated anti-virus software.
4. SOM with k-medoids clustering is a better way in detection of malicious mail virus: with a little false positive rate (2.22%), SOM with k-medoids clustering method shows a high detection rate and overall accuracy in malicious email detection.

In order to find out the capability of detecting new unseen mail viruses with our method, we found some new mail viruses in Internet during our work, which is shown in Table 6, and

tested with all methods we described. These mail viruses were found after May 2003 in our mail server. Its features are confirmed not in our data set. Table 6 shows the testing results of all methods, where anti-virus software Pcc2003 and Norton2003 are updated until May 2003 for fare comparison. From Table 6, we observe that there is still one mail virus that anti-virus software couldn't detect. We may also conclude that:

1. All the mail viruses, in Table 6, must be distributed after 2002: since anti-virus software of 2002 can't detect these entire mail viruses, it must be found in 2003.
2. There would be some mail virus that Pc-cillin and Norton can not detect: from Table 6, Worm-COD has been announced to be a virus by Sophos Anti-virus version 3.37, but Pc-cillin and Norton seem didn't update enough to detect it.
3. There would be some bordline between malicious and benign mails that SOM clustering can't decide: a bordline means it is fuzzy to decide whether Worm-COD is virus or not. An SOM clustering may come to this situation.
4. Naïve Bayes classification may perform better than anti-virus software: Naïve Bayes detect all the viruses surprisingly, may be it needs a further investigation.
5. The clustering approach of our proposed methods, it detected all the viruses, outperformed than the anti-virus software or other methods in the detection of new unseen mail viruses.

Table 6 : Testing results of new email viruses ("√"= detected)

Profile type	PE_BRID.A	WORM_YAHA.G	Worm_COD	LovGate.G	Scram.A
Naïve Bayes	√	√	√	√	√
SOM(k-medoids)	√	√	√	√	√
SOM	√	√	-	√	√
Pcc2003	√	√	-	√	√
Norton2003	√	√	-	√	√
Pcc2002	-	-	-	-	-
Norton2002	-	-	-	-	-

6. Conclusion and future work

The contribution that we presented in this paper was a method for detecting different type of malicious mails included Macro and VBScript's attachments. We have presented a detection model that utilizes Kohonen's self-organizing map and k-medoids to organize mail virus in a domain onto a two-dimensional map. Clearly the proposed method has generated clear clusters, but until we inspect these clusters we will not be able to ascertain if the approach is useful. The organization of the malicious mail is based solely on the mails'

behavior. The resulting map of this system is very meaningful and can be easily incorporated with a mail server to assist detection of malicious emails. Noted that the features extracted were static properties of the mail and did not require executed. Operating from a mail server, the proposed method could automatically filter the email each host receives. All of this could be done without the server's users having to scan attachments themselves or having to download updates for their virus scanners. Furthermore, its evaluation of an attachment is based solely on the behavior of the mail and not the contents of the attachment itself. That added the ability to detect both the set of known malicious mails and a set of previously unseen, but similar malicious mails.

Virus Scanners are updated about every month. 240–300 new malicious executables are created in that time (8–10) a day (White *et. al.* 1999). During this time period, systems are vulnerable to attacks. Our method may catch those new malicious mails without the need for an update. We can implement a network-level email filter that uses our algorithm to catch mail virus before users receive them. We can either wrap the potential malicious viruses or we can block it. If a malicious mail accesses a user's address book and mails copies of itself out over the network, eventually most users of the LAN will clog the network by sending each other copies of the same malicious virus. Stopping the malicious viruses from replicating on a network level would be very advantageous.

The limitation of the current prototype system is that it has only been evaluated on a sample of data. But, this sample has demonstrated that the prototype is effective, and raised issues about pre-processing and dimension reduction needed to tackle larger data sets. Certainly the prototype has been a useful tool for internal purposes, and it is likely to be a useful approach to assist other organizations in better understanding the interests of malicious email virus.

One of the most important areas of future work for this application is the development of more efficient algorithms. The current methods require a machine with a significant amount of memory to generate, and employ the classifiers. Another potential future work of proposed method is to make it into a stand-alone virus scanner and to port the algorithms to different operating systems. Work needs to be done with industry or security sources to develop a standard data set consisting of infected programs, macro and visual basic viruses, and many different sets of benign data. Finally, our future research will be investigating the scalability of the system, so that it can be incorporated with other detection models.

Acknowledgement: The author wants to thanks for the anonymous reviewers' valuable comments. The entire revised paper was further reviewed/improved and proofread based on the comments. Further, the author would like to give thanks to the National Science Council of Taiwan for grant NSC- 92-2218-E-224-015.

References

1. Arnold W. and Tesauro G., Automatically Generated Win32 Heuristic Virus Detection, Proceedings of the 2000 International Virus Bulletin Conference, 2000.
2. Coulthard A. and Vuori T.A., "Computer viruses: a quantitative analysis", Logistics Information Management, vol.15, no.5/6, 2002, pp400-409.
3. Crawford R., Kerchen P., Levitt K., Olsson R., Archer M. and Casillas M., Automated Assistance for Detecting Malicious Code, Proceedings of the 6th International Computer Virus and Security Conference, 1993.
4. Garber L.," Melissa Virus Creates a New Type of Threat", Computer, vol. 32, no.6, 1999, pp16 –19.
5. Gryaznov D., Scanners of the Year 2000: Heuristics, Proceedings of the 5th International Virus Bulletin, 1999.
6. Han J. and Kamber M., Data mining concepts and techniques, USA, Morgan Kaufmann, 2001, pp284-287.
7. Kaufman L. and Rousseeuw P. J., Finding Groups in Data: an Introduction to Cluster Analysis, John Wiley & Sons, 1990,
8. Kephart J. O. and Arnold W. C., "Automatic Extraction of Computer Virus Signatures", 4th Virus Bulletin International Conference, 1994, pp178-184
9. Kerchen P., Lo R., Crossley J., Elkinbard G. and Olsson R.," Static Analysis Virus Detection Tools for UNIX Systems", Proceedings of the 13th National Computer Security Conference, 1990, pp350–365
10. Kohonen T., "The Self-organization maps", Proc. IEEE, vol.78, no.9, 1990, pp1480-1481.
11. Kohonen T., Self-organization map, Springer, 1995,
12. Lee J.S., Hsiang J. and Tsang P.H.," A Generic Virus Detection Agent on the Internet", System Sciences, Proceedings of the Thirtieth Hawaii International Conference on, vol.4, 1997, pp210 –219
13. Lee W., Stolfo S. and Mok K., A Data Mining Framework for Building Intrusion Detection Models, IEEE Symposium on Security and Privacy, 1999.
14. Lo R.W., Levitt K.N. and Olsson R.A., "MCF: a Malicious Code Filter", Computers & Security, vol.14, no.6, 1995, pp541–566.
15. Luke J. and Harris C.J., "The application of CMAC based intelligent agents in the detection of previously unseen computer viruses", Information Intelligence and Systems, Proceedings, 1999 International Conference on, 1999, pp662 –666
16. McGraw G. and Morrisett G., "Attacking malicious code: Report to the infosec research council", IEEE software, vol.17, no.5, 2000, pp33-41.
17. Mendenhall W. and Beaver R., Introduction to Probability and Statistics, Duxbury Press,

1994.

18. Michie D., Spiegelhalter D. J. and Taylor D. C. C., Machine learning of rules and trees; In Machine Learning, Neural and Statistical Classification, Ellis Horwood, 1994.
19. Okamoto T. and Ishida Y., "An Analysis of a Model of Computer Viruses Spreading via Electronic Mail", Systems and Computers in Japan, vol. 33, no.14, 2002, pp81-90.
20. Schultz M. G., Eskin E., Hershkop S. and Stolfo S. J., Data Mining Methods for Detection of New Malicious Executables, in Proceedings of IEEE Symposium on Security and Privacy, IEEE S&P-2001, Oakland, CA: May 14-17, 2001.
21. Schultz M. G., Eskin E., Hershkop S. and Stolfo S. J., MET: An Experimental System for Malicious Email Tracking, in Proceedings of the 2002 New Security Paradigms Workshop, NSPW-2002, Virginia Beach, VA: September 23- 26, 2002.
22. Spafford E. H., The Internet worm program: an analysis, Tech. Report CSD–TR–823, Department of Computer Science, Purdue University, 1988.
23. Symatec's Bloodhound Technology, Understanding Heuristics: Symatec White Paper Series, Vol.XXXIV.
24. Tesauro G., Kephart J. O. and Sorkin G. B., " Neural Networks for Computer Virus Recognition", IEEE Expert, vol.11, no.4, 1996, pp5–6.
25. Trend Micro http://www.trend.com.tw/endusers/products/pcc2000.html
26. White S. R., Swimmer M., Pring E. J., Arnold W. C., Chess D. M. and Morar J. F., "Anatomy of a Commercial-Grade Immune System", IBM Research White Paper, 1999. http://www.av.ibm.com/ScientificPapers/White/Anatomy/anatomy.html.
27. Zenkin D., "Fighting against the Invisible Enemy", Computers & Security, no.20, 2001, pp316-321.