J Comput Virol (2006) 2:163–186
DOI 10.1007/s11416-006-0013-1

ORIGINAL PAPER

Non-signature based virus detection

Towards establishing a unknown virus detection technique using SOM

In Seon Yoo · Ulrich Ultes-Nitsche

Received: 20 February 2006 / Accepted: 12 April 2006 / Published online: 4 July 2006

Abstract A non-signature-based virus detection approach using Self-Organizing Maps (SOMs) is presented in this paper. Unlike classical virus detection techniques using virus signatures, this SOM-based approach can detect virus-infected files without any prior knowledge of virus signatures. Exploiting the fact that virus code is inserted into a complete file which was built using a certain compiler, an untrained SOM can be trained in one go with a single virus-infected file and will then present an area of high density data, identifying the virus code through SOM projection. The virus detection approach presented in this paper has been tested on 790 different virus-infected files, including polymorphic and encrypted viruses. It detects viruses without any prior knowledge – e.g. without knowledge of virus signatures or similar features – and is therefore assumed to be highly applicable to the detection of new, unknown viruses. This non-signature-based virus detection approach was capable of detecting 84% of the virus-infected files in the sample set which included, as already mentioned, polymorphic and encrypted viruses. The false positive rate was 30%. The combination of the classical virus detection technique for known viruses and this SOM-based technique for unknown viruses can help systems be even more secure.

1 Introduction

Virus software is probably the most widely discussed class of computer threat at the moment. To qualify as a virus, a program must meet one particular criterion [1]: the code in the program must be able to replicate or copy itself so as to spread through the infected machine or across to other machines. The main of spreading email viruses in 2005 was in the form of file worms sent via emails. In September 2005, the top ten viruses (53.2%) were file worms distributed by email attachments and/or network shares [2]. Nonetheless, the original method of virus infection, i.e. by infecting a file with a piece of additional code, must not be ignored. These "classical" viruses are the scope of this paper.

Classic virus-detection techniques look for the presence of a virus-specific sequence of instructions, called a virus signature, inside a program. If the signature is found, it is highly probable that the program is infected. In the specific case of searching for a particular malicious code instance, it is not only possible, but performed daily by anti-virus software. Apart from commercial anti-virus solutions for detecting known viruses in virus-infected files, what options are we left with in addressing unknown viruses? The research presented in this paper differs from traditional approaches to the malicious code problem in that it does not attempt to define or identify malicious behavior. Instead, the research focuses on structural characteristics of executable malicious code. This approach allows for methods of examining any application, whether previously known or unknown, in order to determine if it has been tampered with since its original development. Such tampering usually takes the form of an embedded virus or Trojan horse that is activated during subsequent executions of the program.

I. S. Yoo (✉) · U. Ultes-Nitsche
Department of Computer Science,
University of Fribourg, Fribourg, Switzerland
e-mail: in-seon.yoo@unifr.ch

U. Ultes-Nitsche
e-mail: uun@unifr.ch

Prior work presented in Yoo [3] described an approach to visualize virus patterns using self-organizing maps (SOMs) [4]. The SOM visualization of virus-infected files proved that the virus detection approach without prior knowledge of virus signatures using SOM made sense. In this paper, we go into more details of virus recognition and present a virus detection program, *VirusDetector*, which has been developed for determining whether a file is virus-infected or not, based on the prior work using SOM [4].

SOM is a neural network using an unsupervised learning-strategy, which does not require that users specify desired outputs as they would have to in supervised learning [5]. SOM is a so-called feed-forward neural-network whose unsupervised training algorithm incorporates a process called self-organization, i.e. it configures the output units into a topological representation of the original data [6]. (More details can be found in books on SOMs such as [4–8].)

As we will show, virus-infected files cannot hide the presence of the virus through the SOM projection. Exploiting this fact we manage to detect viruses. As no knowledge (no signature, etc) about a virus is required to detect it, not only known viruses but also unknown ones can be detected by our approach. Like defense in depth, we can build much more secure systems by accompanying traditional virus-detection techniques based on virus signatures with our non-signature based virus-detection technique for unknown viruses. This paper explains and presents the non-signature based virus detection approach.

2 Data material and methods

The research presented in this paper focuses only on viruses, not on worms. The initial target file format considered is Microsoft Windows executable format as about 80% of detected viruses or worms were Windows executable files; in particular, approximately 61% had an ".EXE" file extension [9].

2.1 Test data collection

In total, 790 virus-infected files (291 Win9x files and 499 Win32 files), 80 normal (i.e. non-infected) Windows executable files and 15 macro-virus-infected Windows Word files were tested using our approach, including downloadable application programs such as *SSHSecureShell Client-3.2.9.exe*, *dxwebsetup.exe*, *klcodec220b.exe* and already installed executable programs such as *Excel.exe*, *Winword.exe*, *Acrobat.exe*, *servertool.exe*... These virus-infected files were detected and caught in between 1996 and 2004. Even "old" viruses are still of interest as they have several variants which appear nowadays. For example, variants of the CIH/Chernobyl [10] virus have appeared each year since 1998. All test data, file information and the test results are listed in the appendix.

2.2 File format and virus types

Parasitic viruses Parasitic viruses are all viruses which change the content of target files while transferring copies of themselves. The files themselves remain completely or partly usable [11]. The most common method of virus incorporation into a file is by appending the virus to the end of the file as shown in Figure 1. In this process, the virus changes the header of file in such way that the virus code is executed first. In Windows and OS/2 executables (NewEXE – NE, PE, LE, LX), the fields in the NewEXE header are changed. The structure of this header is much more complicated than that of a conventional DOS EXE file, so there are more fields to be changed: the starting address, the number of sections in the file, properties of the sections, etc. In addition to that, before infection, the size of the file may increase to a multiple of one paragraph (16 bytes) in DOS or to a section in Windows and OS/2. The size of the section depends on the properties of the EXE file header [11].

The structure of the data of a virus-infected file is similar to what can be seen in Figure 2, where the positions are octal number. When we examine several virus-infected files, most files have the same size of the DOS stub (say, 128 bytes), and varying other parts. Apart from the virus code, only the PE (portable executable format) header is filled with quite similar character patterns, containing text, data, source and relocation information. The program code and data contain compiler-generated character sequences. Looking at Figure 2, the character sequence of the virus code differs from the other program code, which means that the program code and the inserted virus code have different data characteristics,

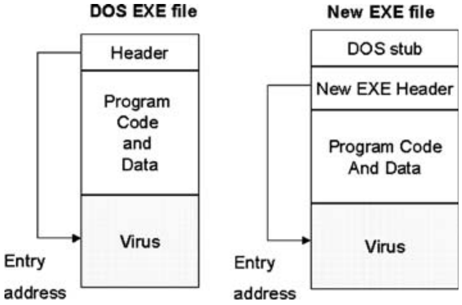
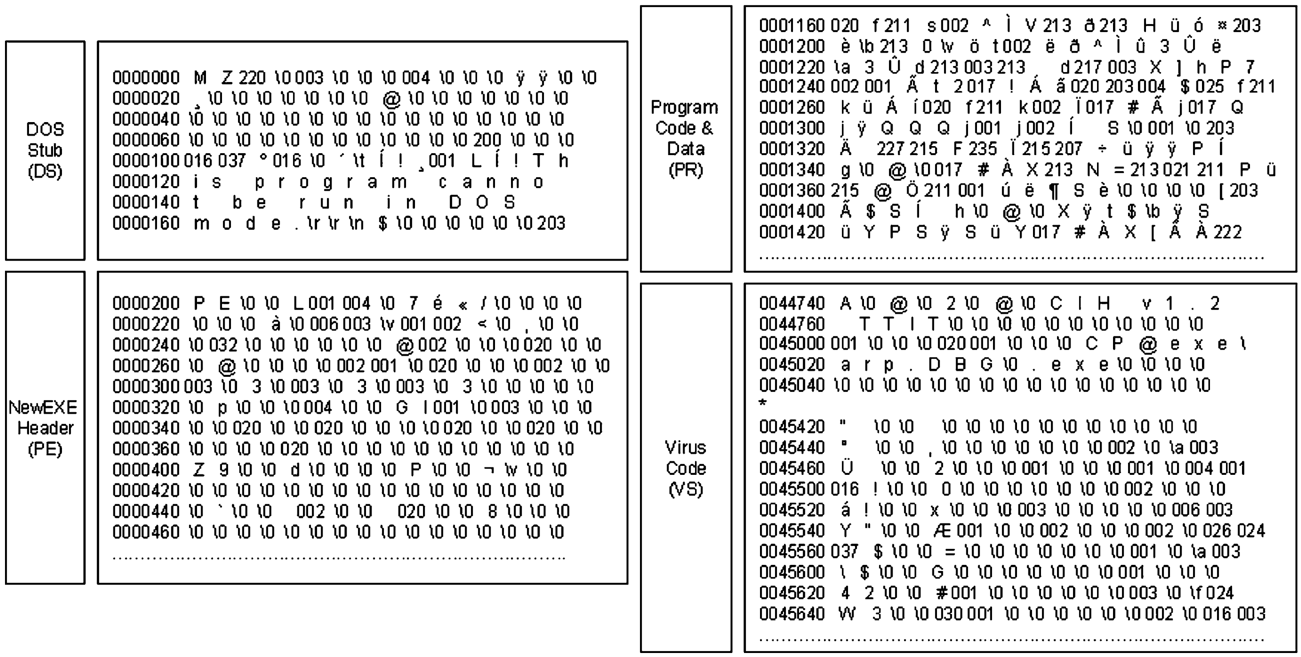


Fig. 1 Virus positions in EXE and document files



```
0000000  M  Z 220 \0 003 \0 \0 \0 004 \0 \0 \0  ÿ  ÿ \0 \0
0000020  ¸ \0 \0 \0 \0 \0 \0 \0  @ \0 \0 \0 \0 \0 \0 \0
0000040 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
0000060 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 200 \0 \0 \0
0000100 016 037  ° 016 \0  ´ \t  Í  !  ¸ 001  L  Í  !  T  h
0000120  i  s     p  r  o  g  r  a  m     c  a  n  n  o
0000140  t     b  e     r  u  n     i  n     D  O  S
0000160  m  o  d  e  . \r \r \n  $ \0 \0 \0 \0 \0 \0 203
```

```
0000200  P  E \0 \0  L 001 004 \0  7  é  «  / \0 \0 \0 \0
0000220 \0 \0 \0  à \0 006 003 \v 001 002  < \0  , \0 \0
0000240 \0 032 \0 \0 \0 \0 \0 \0  @ 002 \0 \0 \0 020 \0 \0
0000260 \0  @ \0 \0 \0 \0 002 001 \0 020 \0 \0 \0 002 \0 \0
0000300 003 \0  3 \0 003 \0  3 \0 003 \0  3 \0 \0 \0 \0 \0
0000320 \0  p \0 \0 \0 004 \0 \0  G  I 001 \0 003 \0 \0 \0
0000340 \0 \0 020 \0 \0 020 \0 \0 \0 \0 020 \0 \0 020 \0 \0
0000360 \0 \0 \0 \0 020 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
0000400  Z  9 \0 \0  d \0 \0 \0 \0  P \0 \0  ¬ \v \0 \0
0000420 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
0000440 \0  ` \0 \0    002 \0 \0    020 \0 \0  8 \0 \0 \0
0000460 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
..........................................................
```

```
0001160 020  f 211  s 002  ^  Ì  V 213  ð 213  H  ü  ó  ¤ 203
0001200  è \b 213  0 \v  ö  t 002  ë  ð  ^  Ì  û  3  Û  ë
0001220 \a  3  Û  d 213 003 213     d 217 003  X  ]  h  P  7
0001240 002 001  Ã  t  2 017  !  Á  ã 020 203 004  $ 025  f 211
0001260  k  ü  Á  í 020  f 211  k 002  Ï 017  #  Ã  j 017  Q
0001300  j  ÿ  Q  Q  Q  j 001  j 002  Í     S \0 001 \0 203
0001320  Ä     227 215  F 235  Ï 215 207  ÷  ü  ÿ  ÿ  P  Í
0001340  g \0  @ \0 017  #  À  X 213  N  = 213 021 211  P  ü
0001360 215  @  Ö 211 001  ú  ë  ¶  S  è \0 \0 \0 \0  [ 203
0001400  Ã  $  S  Í     h \0  @ \0  X  ÿ  t  $ \b  ÿ  S
0001420  ü  Y  P  S  ÿ  S  ü  Y 017  #  À  X  [  Å  À 222
..........................................................
```

```
0044740  A \0  @ \0  2 \0  @ \0  C  I  H     v  1  .  2
0044760     T  T  I  T \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
0045000 001 \0 \0 \0 020 001 \0 \0 \0  C  P  @  e  x  e  \
0045020  a  r  p  .  D  B  G \0  .  e  x  e \0 \0 \0 \0
0045040 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
*
0045420  "    \0 \0    \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
0045440  º    \0 \0  , \0 \0 \0 \0 \0 \0 \0 002 \0 \a 003
0045460  Ü    \0 \0  2 \0 \0 \0 001 \0 \0 \0 001 \0 004 001
0045500 016  ! \0 \0  0 \0 \0 \0 \0 \0 \0 \0 002 \0 \0 \0
0045520  á  ! \0 \0  x \0 \0 \0 003 \0 \0 \0 \0 \0 006 003
0045540  Y  " \0 \0  Æ 001 \0 \0 002 \0 \0 \0 002 \0 026 024
0045560 037  $ \0 \0  = \0 \0 \0 \0 \0 \0 \0 001 \0 \a 003
0045600  \  $ \0 \0  G \0 \0 \0 \0 \0 \0 \0 001 \0 \0 \0
0045620  4  2 \0 \0  # 001 \0 \0 \0 \0 \0 \0 003 \0 \f 024
0045640  W  3 \0 \0 030 001 \0 \0 \0 \0 \0 \0 002 \0 016 003
..........................................................
```

Fig. 2 Structure of Windows Executable file: DOS Stub, PE header part, Program Code and Data and Virus position

e.g. because the original code was compiled as one solid piece of code, and the virus is injected only afterwards.

If these four different areas in the virus-infected file are identified, they can be used to prove the virus visualization is correct as done in the next section. Each different area will be labeled, e.g. DS for the DOS stub, PE for the NewEXE header, PR for the program code and data, and VS for the virus code. These labels are only used to identify where each part of the file will be located in the SOM projection, in order to show that the virus code is located in an area of close neighborhood; the labels are not necessary for SOM training and visualization.

Macro viruses Another virus type appearing in Windows systems is the macro virus. Macro viruses are programs written in macro languages of programs such as Microsoft Word and Excel as presented in Figure 3. To propagate, such viruses use the capabilities of macro languages and with their help transfer themselves from one infected file, e.g. document or spreadsheet, to another. Microsoft Word Version 6 and 7 allows to encrypt macros in documents [12]. Therefore, some word viruses are present inside the infected documents in an encrypted, execute-only form.

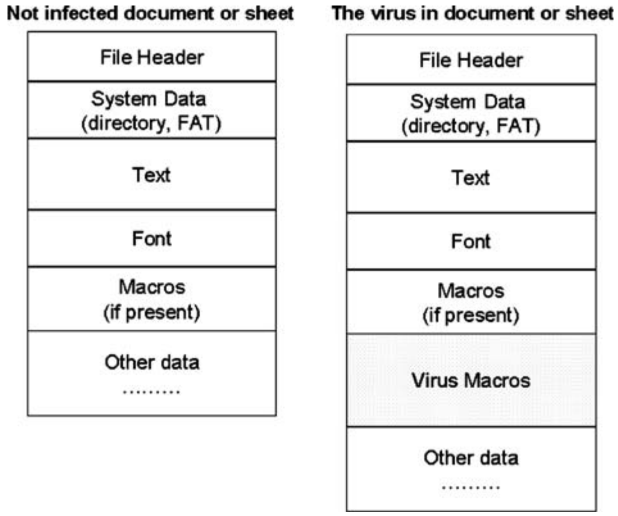


Fig. 3 Macro virus position in an infected document

Polymorphic viruses Polymorphic viruses cannot, or can with significant effort, be detected using virus signatures. Polymorphic viruses try to remain undetected by changing their structure with each infection. There is no unique signature, that anti-virus programs can search for. Some polymorphic viruses even use different encryption techniques with every infection. In this paper, we do not deal with polymorphic viruses separately because these polymorphic viruses can be included in parasitic and macro viruses, and they can be handled similarly to parasitic and macro viruses with our approach. We simply assume that polymorphic viruses are somehow inserted in the executable file and this inserted virus code, as described in Section 2.2, will differ again from the original program code. Thus whether encrypted or polymorphic, the virus code can be distinguished from

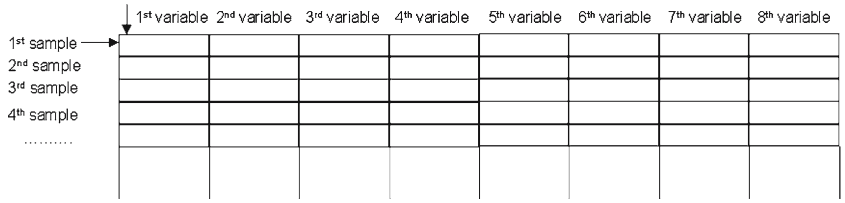


Fig. 4 Table-format data: there can be any number of samples, but all samples have fixed length, and consist of the sample variables

the original program code as it is injected into an intact, kind of homogeneous program file.

2.3 SOM training and visualization

The SOM Toolbox 2.0 [13], a software library for MATLAB 6.0 [14], was used under Linux to visualize virus-infected files. The visualization experiments were carried out under Linux as a precaution against infection with any of the viruses we used (they were all Windows viruses). A virus detection program, *VirusDetector*, was then developed which uses the SOM algorithms initially employed in the MATLAB visualization. However, it does not visualize the viruses anymore but solely decides whether or not the visualization had contained the so-called *virus mask* which indicates the presence of a virus.

Data preparation for SOM training To train a SOM, a virus-infected file's binary data was converted into a table of numerical values. In general SOM data, each row of the table is one data sample, which means the entire table consists of n different data samples. The columns of the table are the variables. The items in one row are values of these variables from the data set. The number of variables depends on data features and was chosen to be eight in our experiments.

The eight variables are presented in Figure 4. For the virus-detecting SOM, multiple bytes were given to each variable to reflect characteristic data features. Each row of the table is one row of one (possibly virus-infected) file's binary data, and the table is a transformed form of the (possibly virus-infected) file ("file under test").

To match the table structure with multiple bytes per variable in the table, a common octal-dump open source program (command name "od" in Linux) was rewritten to remove the front offset information from the dump output, transforming a binary file into a short-integer typed data format. The short integer format was chosen in order to keep the range of numerical values relatively small (in C, short integers range from −32,768 to +32,767). In this transformation, 4 bytes are assigned to each of the eight variables per row, i.e. each input sample of the SOM will contain 32 bytes of the file under test. To summarize, the table consists of single 8 × 4-byte data samples representing n (number of rows) different portions of the file under test without overlapping data. It should be noted that this is an unusual way of using a SOM (it is not trained with n different data samples but it is "trained" with n fractions of the same sample).

Visualization method There are many different methods of displaying SOMs. We use the unified distance matrix or Umatrix since the shape of the Umatrix corresponds to the density structure of the input data, and the location of the best-matching prototypes corresponds to the topography of the input data. Therefore, the SOM "trained" with a file under test reflects the file's structure in the Umatrix.

The Umatrix represents the map as a regular grid of neurons which can be visualized easily. Every neuron gets a numeric value assigned that corresponds to its local density in the input-space: the average distance between its prototype and the neighboring nodes' prototypes. A low value corresponds to a high local density, a high value to a low local density. For visualization purposes, these values can be converted easily into a color scheme: a light (low value) color corresponds to a high local density, a dark (high value) color to a low local density, or vice versa.

Even though a colorful visualization was produced easily using MATLAB, *VirusDetector* does not require colorful visualization; only the Umatrix neurons' values are used for detection. Thus, *VirusDetector* uses an internal Black/White representation of the Umatrix's node values using textual information.

Process of SOM training and visualization Figure 5 illustrates the process of SOM training and visualization. Each step in Figure 5 is described subsequently.

1. Data buffer represents the input data in a table format. The size of the input data is arbitrary. In this example we assume we can partition it into four parts labeled A, B, C and D, each representing n rows of the input data (the entire input data consists of $4n$ rows, where n is an arbitrary number). A consists of n rows containing the data designated by $a_{11}, a_{12}, \ldots, a_{18}, a_{21}, a_{22}, \ldots, a_{28}, \ldots$, and

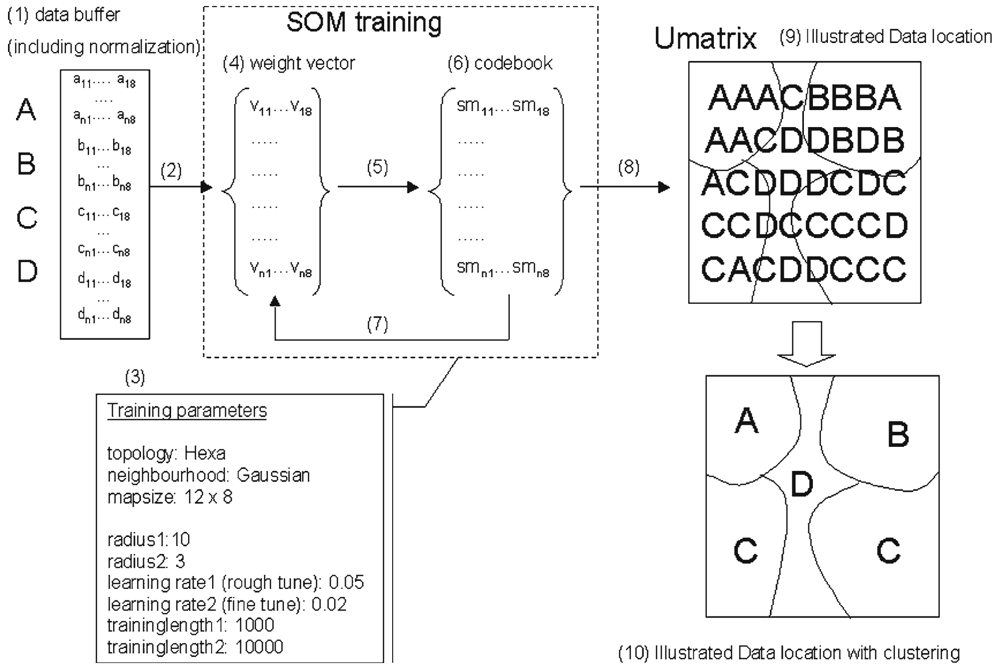


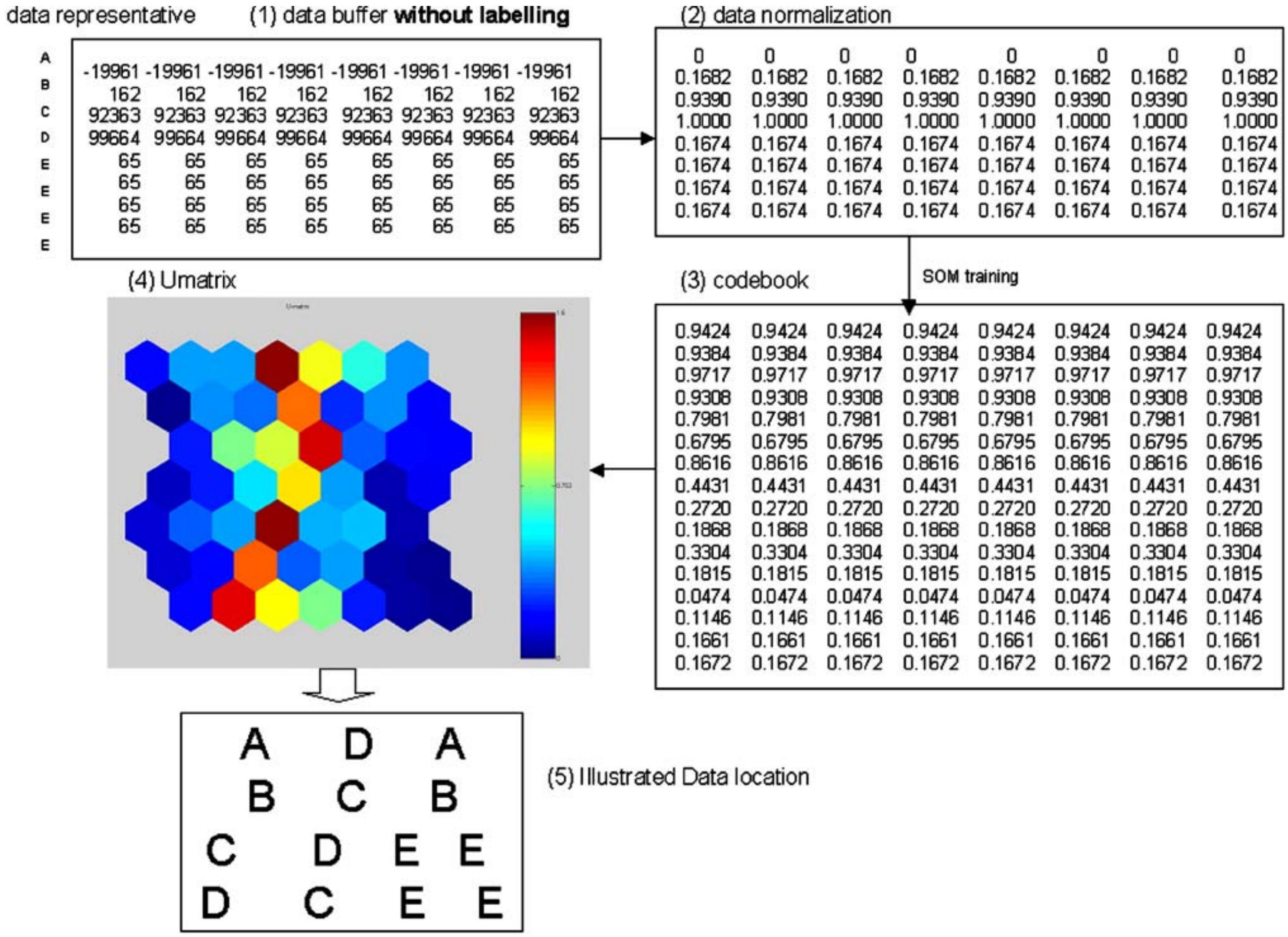
Fig. 5 Process of SOM training and visualization

$a_{n1}, a_{n2}, \ldots, a_{n8}$. Similarly B, C, and D are defined. There is no data overlap between any of these parts. Finally, in this step, the entire data is normalized and then used to train the SOM.

2. At the beginning of the SOM training, the entire data is quantized and in an initial training step, eigenvectors to each entity are calculated. There exists a random and a linear initialization phase in the training of SOMs. In our approach, we use the linear initialization, which uses a linear mapping whose eigenvectors are used in the initialization phase.
3. Using certain parameters in the SOM training phase, namely hexagonal topology, Gaussian neighborhood and particular map size values, each row value of the weight vector is calculated and updated until finding best matches to the input data.
4. The weight vector is used for updating the vector value in each point (SOM cell).
5. Once the entire weight vector structure is calculated, the values are saved in the so-called codebook vector. In each step, the winning entry is found in the codebook using Euclidean distance by going through the list of all weight vectors, each time computing the distance between the codebook and the input entry from the weight vector.
6. Once the fine tune is performed, the codebook is fixed and consists of best matched vector values.
7. The weight vector and the codebook are referenced and updated for rough tune and fine tune.
8. The Umatrix is one of the methods to visualize a SOM. The Umatrix visualizes the codebook vector values.
9. The Umatrix visualization reflects the quantized data. Similar data is located in a close neighborhood to one another, producing an area of similar data density.
10. Having given different sections of the data with different names allow a represention the location of each group of the data in the Umatrix. The purpose of the illustration is to cluster around the data in the Umatrix, about which we know the initial in the input data, aiming at identifying what the SOM does to that data. This cluster of data represents the data's close neighborhood. In this way, virus data will turn out to be represented in close neighborhood to one another, which can be distinguished from the other data.

SOM projection Figure 6 shows a simple example to illustrate the process of SOM training and visualization. It shows that if there is similar "isolated" data in a file, it will be distinguished from remaining data during the training of the SOM.

1. There is a very small input data set in the data buffer. For simplicity, we label identical data rows with one of the letters A, B, C, D and E. A, B, C, and D represent exactly one row of data,



(1) data buffer **without labelling**

A	-19961	-19961	-19961	-19961	-19961	-19961	-19961	-19961
B	162	162	162	162	162	162	162	162
C	92363	92363	92363	92363	92363	92363	92363	92363
D	99664	99664	99664	99664	99664	99664	99664	99664
E	65	65	65	65	65	65	65	65
E	65	65	65	65	65	65	65	65
E	65	65	65	65	65	65	65	65
E	65	65	65	65	65	65	65	65

(2) data normalization

0	0	0	0	0	0	0	0
0.1682	0.1682	0.1682	0.1682	0.1682	0.1682	0.1682	0.1682
0.9390	0.9390	0.9390	0.9390	0.9390	0.9390	0.9390	0.9390
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
0.1674	0.1674	0.1674	0.1674	0.1674	0.1674	0.1674	0.1674
0.1674	0.1674	0.1674	0.1674	0.1674	0.1674	0.1674	0.1674
0.1674	0.1674	0.1674	0.1674	0.1674	0.1674	0.1674	0.1674
0.1674	0.1674	0.1674	0.1674	0.1674	0.1674	0.1674	0.1674

(3) codebook

0.9424	0.9424	0.9424	0.9424	0.9424	0.9424	0.9424	0.9424
0.9384	0.9384	0.9384	0.9384	0.9384	0.9384	0.9384	0.9384
0.9717	0.9717	0.9717	0.9717	0.9717	0.9717	0.9717	0.9717
0.9308	0.9308	0.9308	0.9308	0.9308	0.9308	0.9308	0.9308
0.7981	0.7981	0.7981	0.7981	0.7981	0.7981	0.7981	0.7981
0.6795	0.6795	0.6795	0.6795	0.6795	0.6795	0.6795	0.6795
0.8616	0.8616	0.8616	0.8616	0.8616	0.8616	0.8616	0.8616
0.4431	0.4431	0.4431	0.4431	0.4431	0.4431	0.4431	0.4431
0.2720	0.2720	0.2720	0.2720	0.2720	0.2720	0.2720	0.2720
0.1868	0.1868	0.1868	0.1868	0.1868	0.1868	0.1868	0.1868
0.3304	0.3304	0.3304	0.3304	0.3304	0.3304	0.3304	0.3304
0.1815	0.1815	0.1815	0.1815	0.1815	0.1815	0.1815	0.1815
0.0474	0.0474	0.0474	0.0474	0.0474	0.0474	0.0474	0.0474
0.1146	0.1146	0.1146	0.1146	0.1146	0.1146	0.1146	0.1146
0.1661	0.1661	0.1661	0.1661	0.1661	0.1661	0.1661	0.1661
0.1672	0.1672	0.1672	0.1672	0.1672	0.1672	0.1672	0.1672

Fig. 6 Example of SOM training and visualization

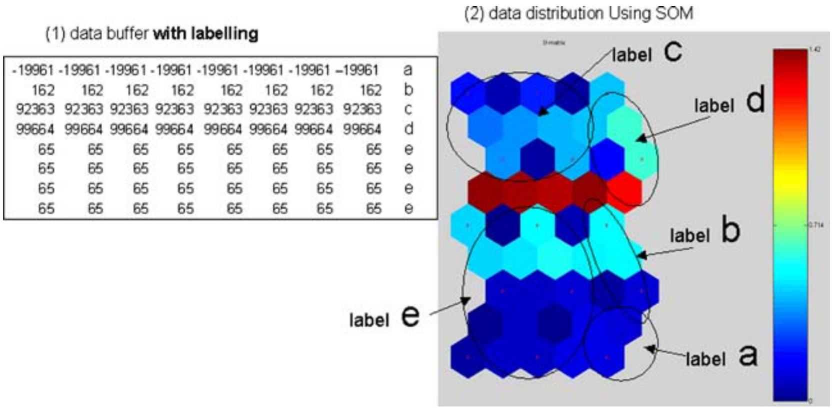
E represents the last four rows of identical data. We will use this example to present how the Umatrix reflects the existence the area of identical data labeled with E.

2. This is the input data after normalization. The entire data structure is not changed – E still represents rows of identical data.
3. After the SOM training process, the codebook is produced as presented in Figure 6 (3).
4. The Umatrix presents the codebook vector values. In our example, only the bottom-right neurons are in close neighborhood to one another when compared to other neurons.
5. The illustration presents how SOM training located the data based on their density. The "E data" appears in an area of neurons with a close neighborhood relation to one another compared to other data because relatively many identical data instances (labeled with E) were fed into the SOM to train it. That caused the SOM to calculate a high density of data belonging to E, meaning that the neurons are in close neighborhood to one another. The illustrated data location using the labels was created after identifying each neuron in the Umatrix with its original label. We will call the way the original data is distributed in the UMatrix the "SOM distribution" (explained in the next section using Figure 7) with data labels.

It is important to note that we are using labels only to identify where a previously identified part of the input data will be distributed. They are not a part of the training process itself. By using such a labeling we will show that an area in the UMatrix, which we will call the virus mask, indeed represents virus data. In the detection system we develop, data is obviously unlabeled as we do not have any prior knowledge about where, if at all, virus data and other fraction, of the data are located.

SOM data distribution This technique was used to identify which data part was located where. Using this SOM distribution, the virus part was identified and proved

Fig. 7 Example of SOM distribution with labeled data



(1) data buffer **with labelling**

-19961	-19961	-19961	-19961	-19961	-19961	-19961	–19961	a
162	162	162	162	162	162	162	162	b
92363	92363	92363	92363	92363	92363	92363	92363	c
99664	99664	99664	99664	99664	99664	99664	99664	d
65	65	65	65	65	65	65	65	e
65	65	65	65	65	65	65	65	e
65	65	65	65	65	65	65	65	e
65	65	65	65	65	65	65	65	e

(2) data distribution Using SOM

that the virus detection approach using SOM was reasonable. Note that this SOM distribution was only used for the purpose of proving where input data was distributed to.

The way to produce this distribution of data as presented in Figure 7 is a bit different from normal SOM training. That is the reason why the output of the Umatrix [Figure 7 (2)] is different from Figure 6 (4).

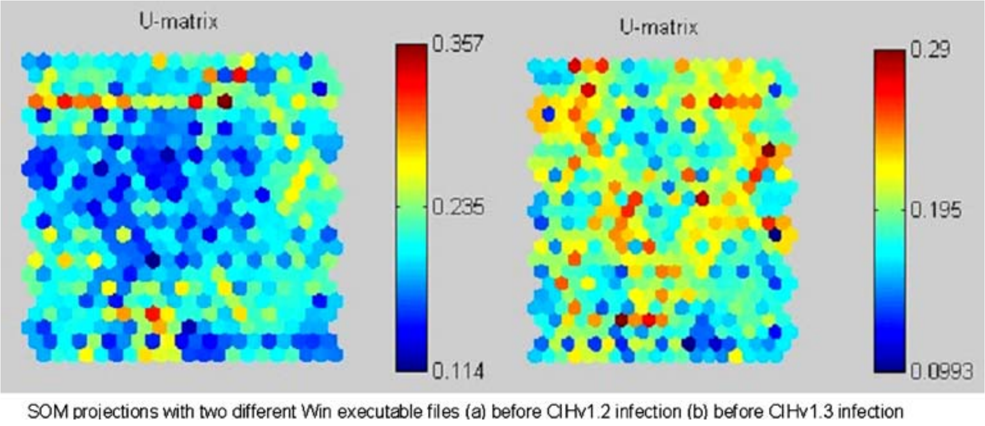
1. The same data buffer, which was used in Figure 6, holds labeled data at the end of each row. In this case, each row belongs to that label, e.g. the first row belongs to label ‘a’.
2. This SOM distribution represents each label’s data density. Each label’s data is located together. This method is used for identifing each data’s location and density. The SOM distribution proves that the E part is located together in Figure 6 (4) and illustrated in Figure 6 (5).

2.4 Virus visualization

Here we discuss how viruses can be visualized using our approach. For the two types of viruses – parasitic viruses and macro viruses – one example will be presented. In the SOM projection, we will identify a particular pattern which signals the presence of a virus. We will refer to this pattern as the *virus mask*.

Win95.CIH virus We discuss this virus here as it was one of the most famous viruses which appeared periodically from 1998 to 2004 in slightly different variants. Figure 8 shows the SOM projection of two Windows’ executable files before Win95.CIH infection. The SOM projections of the tested Windows executables are different to one another, because they are different executable files. However, after Win95.CIH infection, the SOM projections of the files develop a similar pattern, as presented in Figure 9: a dark area (navy colored in the colored map, representing an area of SOM cells where each cell has a short distance to its neighboring cells). The easily identifiable dark spot is what we call the virus mask. It is the pattern that signals the presence of a virus in the file under test.

Figure 9 shows two SOM projections after training the SOM with two Win95.CIH (versions 1.2 and 1.3) infected Windows’ executable files respectively. Each Win95.CIH/Chernobyl virus mask has an obvious



SOM projections with two different Win executable files (a) before CIHv1.2 infection (b) before CIHv1.3 infection

Fig. 8 SOM projections of two different Windows EXE files before Win95.CIH virus infection

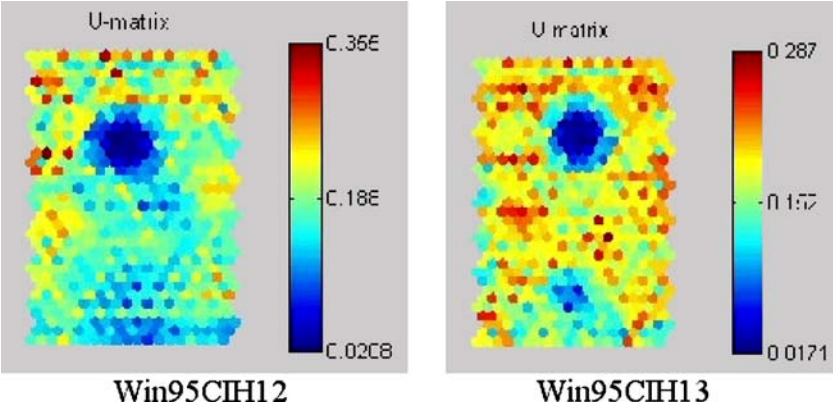


Fig. 9 SOM projections of Windows EXE files infected by Win95.CIH v1.2 and v1.3 viruses

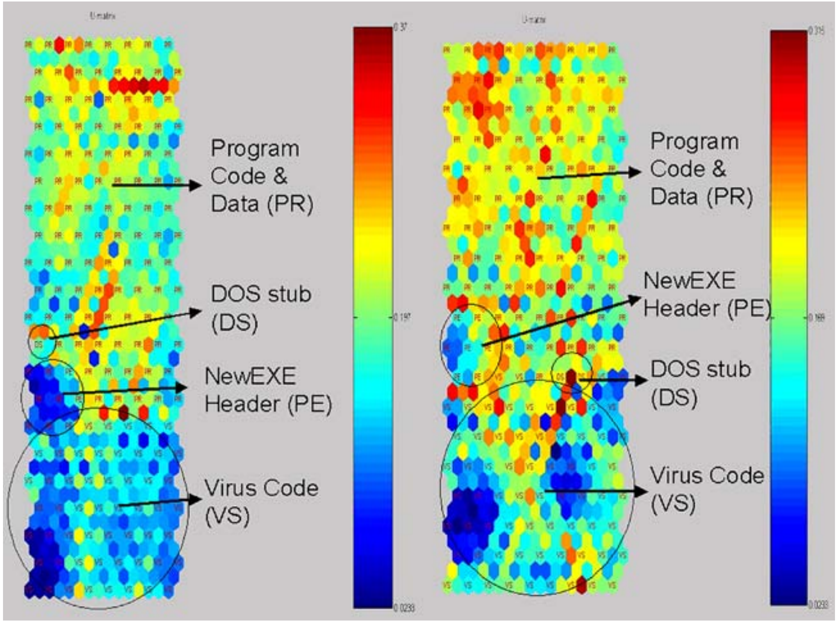


Fig. 10 Virus SOM Distribution of CIH 1.2 and 1.3 viruses

location: top of the center. Although the tested Windows executable files were different, the SOM projections of CIH virus-infected files look similar and have the same sort of projection map.

To prove that the virus mask indeed represents the CIH virus code, the SOM was trained in another experiment with data to which labels DS (DOS stub), PE (NewEXE header), PR (program code) and VS (virus code) were attached. The labels reflect the structure of an infected file as was presented in Figure 2. In order to attach the labels, the structure of the infected file was examined “by hand”. When the data set was produced, to each row a label was added, as shown in Figure 7 (1). The result of the SOM distribution with labeled data is presented in Figure 10. This SOM distribution has grouped the same labels together. Therefore the figures of the SOM distribution with labels are quite different from the normal SOM projection (Figure 9).

The circles in Figure 10 identify the different data areas except for the program code area (PR). PR is simply represented by the remaining area. As Figure 10 shows, there are two parts where cells have a smaller distance to their neighboring cells: PE (NewEXE header) and VS (Virus Code). Even though PE shows an area of smaller distances between two SOM cells, VS dominates the area of small cell distances (black area in gray-scale, navy-colored area in color print), proving that the virus mask identified in Figure 9 represents indeed virus code.

MacroWord97.Mbug Virus When the SOM was trained with MacroWord97.Mbug viruses, the results were like that in Figure 11. Although the figures of the SOM projections look very similar to the virus mask in the CIH example, the result of analyzing the SOM data distribution (Figure 12) showed that the majority of the top-center located data was not the virus code (VS) part but were labeled OD and SD (these reflect the particular structure of Word documents and will not be discussed any further in this paper).

Thus, the result indicates that we cannot deal with macro viruses in the same way we deal with parasitic viruses. Our approach, therefore is insensitive to macro viruses. (Since it is sensitive to parasitic viruses it is still worth being employed whatever.)

2.5 The process of virus detection

With our knowledge about the virus mask in SOM projections of virus-infected files, we built *VirusDetector* which, based on SOM algorithms, detects the virus mask. Apart from the colorful visualization result using MATLAB, *VirusDetector* uses a simple Black and White color scheme.

Figure 13 shows the step-wise process in *VirusDetector*. After feeding a binary executable file to *VirusDetector*, the data is converted into integer format. Then the data is normalized using certain SOM parameters such as topology type: Hexa, neighborhood: Gaussian, map size: 12 × 8, radius1: 10, radius2: 3, learning rate1: 0.05 for rough tune, learning rate2: 0.03 for fine tune, trainlength1: 1,000, and trainlength2: 10,000. Afterwards, the SOM is trained with the normalized data. Unfortunately, there is no theoretical basis for the selection of these parameters [4]. The parameter settings were determined experimentally from our visualization results using the SOM toolbox in MATLAB. Note that the graphical representation of the SOM depends on the initialization, meaning that a virus mask might be located elsewhere, or be shown in a totally undetectable form for a different initialization. However, using the mentioned (good) parameters, the SOM projection produces the patterns which *VirusDetector* can search for and finally find the virus mask if it is present.

While the SOM is trained, it produces a codebook for storing the data. Using this codebook, *VirusDetector* calculates the Umatrix in Black and White. High values (dark SOM cells) indicate high data density which is particularly the case for virus data. A "factor value" is used for selecting which high values are significant. According to these B/W Umatrix values, *VirusDetector* filters out the Umatrix values above the factor value and saves them. To represent the filtered-out values on the two dimensional projection plane, the character "S" is used (each SOM cell with an assigned gray-scale value above the factor value is represented by an "S", all other cells are blank). This produces a representation of the projection plane in form of ASCII strings. After producing the map of "S" s, *VirusDetector* searches for the virus pattern and marks the virus mask if present. After identifying the virus mask, *VirusDetector* decides that the file under test is virus-infected.

Here is an example of the detection process as depicted in Figure 14.

1. Data buffer presents a virus-infected file's short-integer-formatted data.
2. After normalization in *VirusDetector*, the codebook of the input data is produced.
3. *VirusDetector* calculates the values of the Umatrix neurons from the codebook, assigning a gray-scale color value to each neuron (SOM cell).
4. If the factor value is 72, which is used in *VirusDetector*, the neurons with a higher value than the factor value are selected, as presented in Figure 15a.
5. The filtered values are replaced by character "S" to create strings as presented in Figure 15b, and *VirusDetector* manipulates the strings to search for the virus pattern. In the final step, *VirusDetector* decides on the found pattern whether or not it represents a virus mask.

3 Results

Using *VirusDetector*, we tested the 790 virus-infected files listed in the appendix. The test set only includes virus-infected executable Windows files. Since experiments with labeling the input data showed that it cannot detect macro viruses properly, macro viruses are excluded from the test set.

3.1 Unencrypted parasitic viruses

Figure 16 presents the SOM projection of the Win95. Anxiety virus and recognition result produced by *VirusDetector* respectively. As the result shows, the virus mask is represented roughly by the strings of "S" s. Figure 17 contains the equivalent result for the Win32.HLLP. Semisoft virus. Tables 1 and 2 summarize the experiments conducted on some virus-infected executables in Win9x and Win32 format, respectively. Results on the recognition of non-infected executable files are presented in Table 3. The complete result list can be found in the appendix.

3.2 Polymorphic and encrypted parasitic viruses

Among the 790 virus-infected test files, there were 30 Win9x encrypted and 15 polymorphic parasitic viruses. These represent about 15% of the tested Win9x virus-infected files. In the case of Win32 executables, 30 were

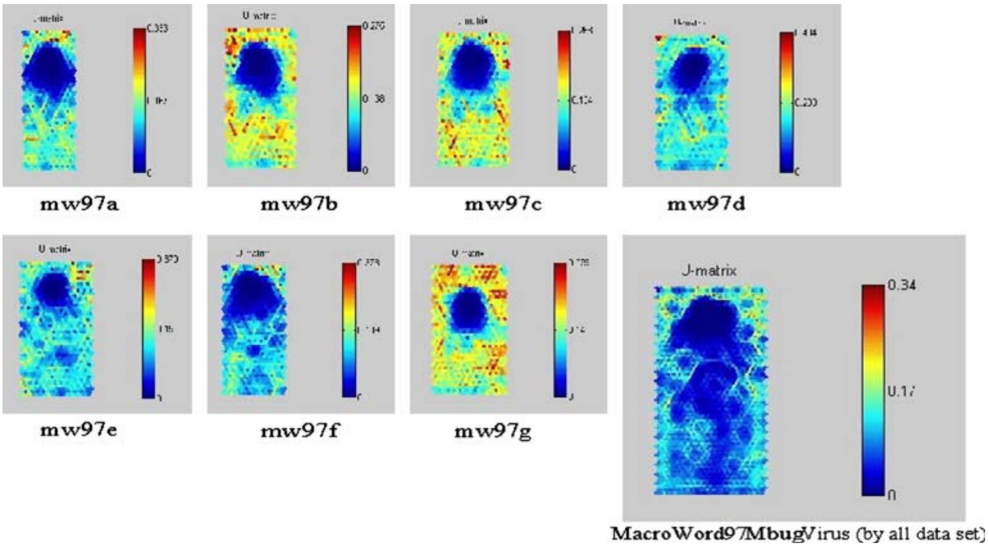
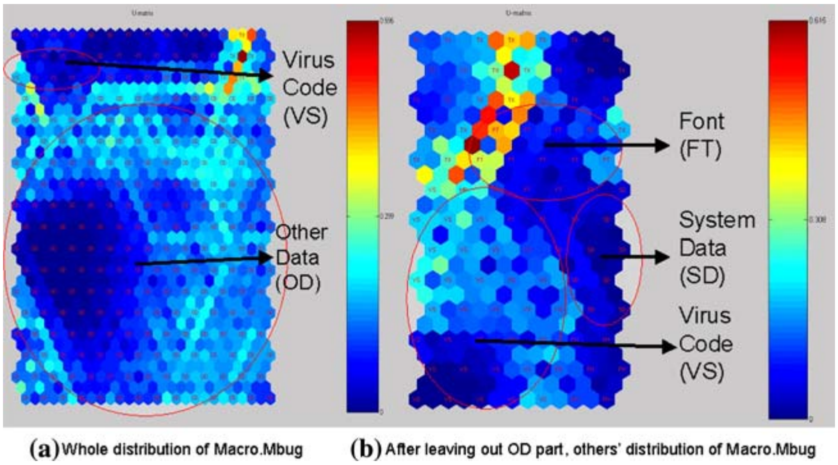
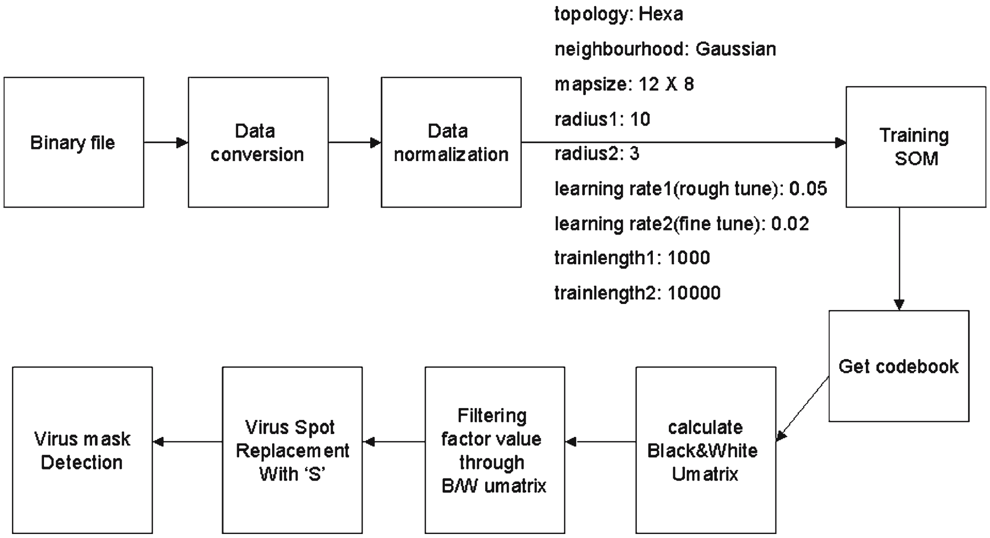


Fig. 11 SOMs of Word97 file infected by Macro viruses



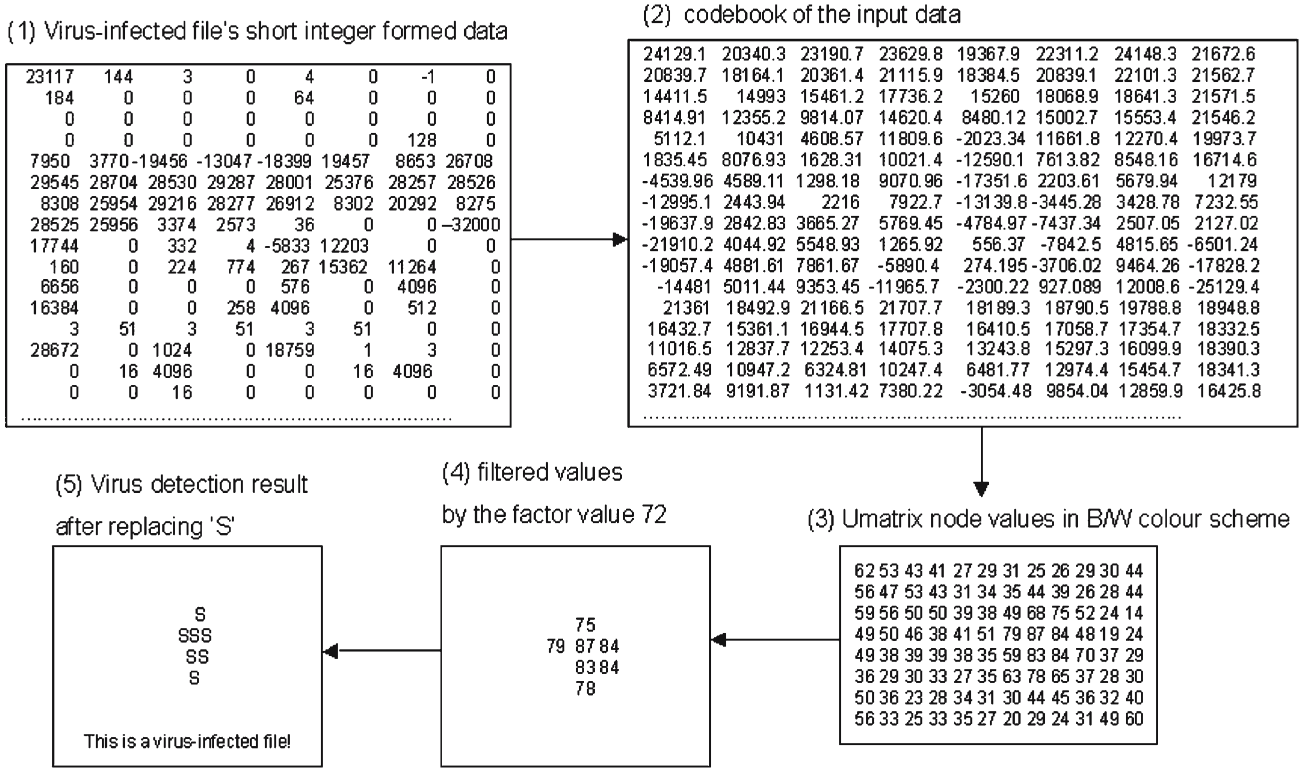
(a) Whole distribution of Macro.Mbug (b) After leaving out OD part, others' distribution of Macro.Mbug

Fig. 12 SOM distribution of Word97 file infected by Macro viruses, *left* whole file projection, *right* without having Other Data part


Binary file
Data conversion
Data normalization
Training SOM
Get codebook
calculate Black&White Umatrix
Filtering factor value through B/W umatrix
Virus Spot Replacement With 'S'
Virus mask Detection

topology: Hexa
neighbourhood: Gaussian
mapsize: 12 X 8
radius1: 10
radius2: 3
learning rate1(rough tune): 0.05
learning rate2(fine tune): 0.02
trainlength1: 1000
trainlength2: 10000

Fig. 13 The process of virus detection in *VirusDetector*



(1) Virus-infected file's short integer formed data

```
23117    144      3      0      4      0     -1      0
  184      0      0      0     64      0      0      0
    0      0      0      0      0      0      0      0
    0      0      0      0      0      0    128      0
 7950   3770-19456 -13047-18399 19457   8653 26708
29545  28704  28530  29287  28001  25376  28257  28526
 8308  25954  29216  28277  26912   8302  20292   8275
28525  25956   3374   2573     36      0      0 –32000
17744      0    332      4  -5833  12203      0      0
  160      0    224    774    267  15362  11264      0
 6656      0      0      0    576      0   4096      0
16384      0      0    258   4096      0    512      0
    3     51      3     51      3     51      0      0
28672      0   1024      0  18759      1      3      0
    0     16   4096      0      0     16   4096      0
    0      0     16      0      0      0      0      0
 ...........................................................
```

(2) codebook of the input data

```
 24129.1  20340.3  23190.7  23629.8  19367.9  22311.2  24148.3  21672.6
 20839.7  18164.1  20361.4  21115.9  18384.5  20839.1  22101.3  21562.7
 14411.5    14993  15461.2  17736.2    15260  18068.9  18641.3  21571.5
 8414.91  12355.2  9814.07  14620.4  8480.12  15002.7  15553.4  21546.2
  5112.1    10431  4608.57  11809.6 -2023.34  11661.8  12270.4  19973.7
 1835.45  8076.93  1628.31  10021.4 -12590.1  7613.82  8548.16  16714.6
-4539.96  4589.11  1298.18  9070.96 -17351.6  2203.61  5679.94    12179
-12995.1  2443.94     2216   7922.7 -13139.8 -3445.28  3428.78  7232.55
-19637.9  2842.83  3665.27  5769.45 -4784.97 -7437.34  2507.05  2127.02
-21910.2  4044.92  5548.93  1265.92   556.37  -7842.5  4815.65 -6501.24
-19057.4  4881.61  7861.67  -5890.4  274.195 -3706.02  9464.26 -17828.2
  -14481  5011.44  9353.45 -11965.7 -2300.22  927.089  12008.6 -25129.4
   21361  18492.9  21166.5  21707.7  18189.3  18790.5  19788.8  18948.8
 16432.7  15361.1  16944.5  17707.8  16410.5  17058.7  17354.7  18332.5
 11016.5  12837.7  12253.4  14075.3  13243.8  15297.3  16099.9  18390.3
 6572.49  10947.2  6324.81  10247.4  6481.77  12974.4  15454.7  18341.3
 3721.84  9191.87  1131.42  7380.22 -3054.48  9854.04  12859.9  16425.8
 .............................................................................
```

(3) Umatrix node values in B/W colour scheme

```
62 53 43 41 27 29 31 25 26 29 30 44
56 47 53 43 31 34 35 44 39 26 28 44
59 56 50 50 39 38 49 68 75 52 24 14
49 50 46 38 41 51 79 87 84 48 19 24
49 38 39 39 38 35 59 83 84 70 37 29
36 29 30 33 27 35 63 78 65 37 28 30
50 36 23 28 34 31 30 44 45 36 32 40
56 33 25 33 35 27 20 29 24 31 49 60
```

(4) filtered values by the factor value 72

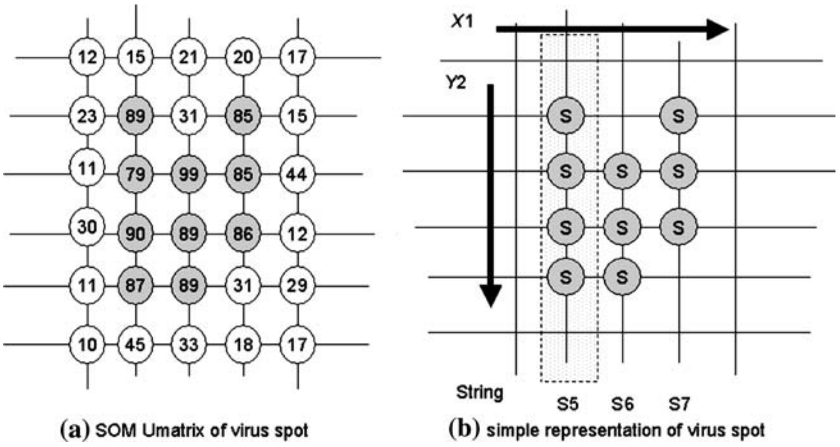
```
      75
   79 87 84
      83 84
      78
```

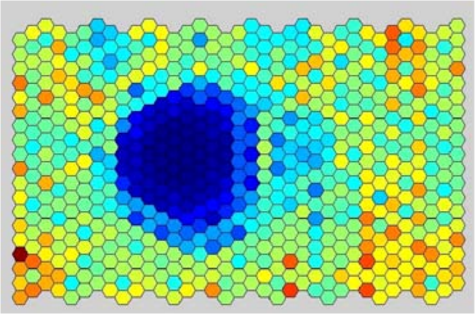
(5) Virus detection result after replacing 'S'

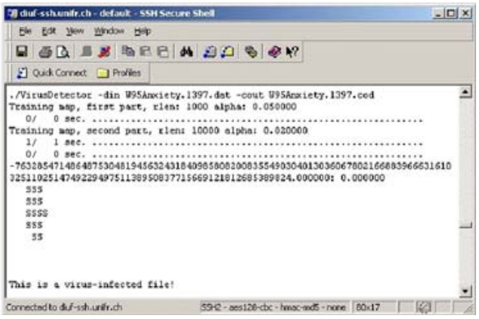
```
   S
  SSS
   SS
   S

This is a virus-infected file!
```

Fig. 14 Virus detection example

Fig. 15 In B/W Umatrix, only bigger values than the factor value 72 are selected and replaced by character 'S'

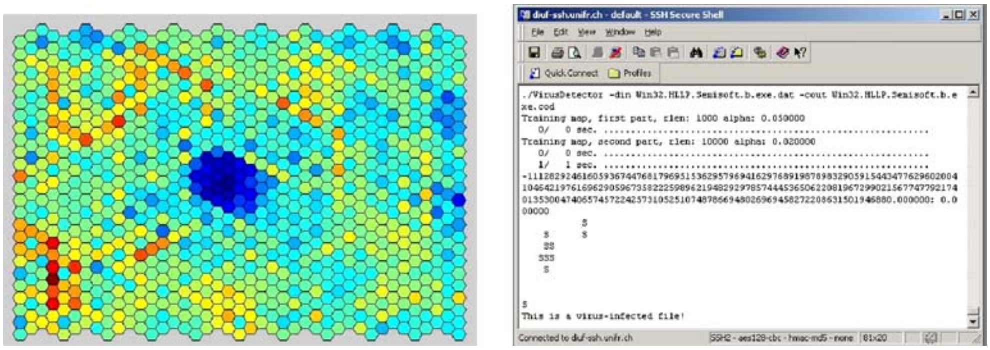





duf-ssh.unifr.ch - default - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
Connected to duf-ssh.unifr.ch
SSH2 - aes128-cbc - hmac-md5 - none
80x17

```
./VirusDetector -din W95Anxiety.1397.dat -cout W95Anxiety.1397.cod
Training map, first part, rlen: 1000 alpha: 0.050000
   0/    0 sec. ...........................................................
Training map, second part, rlen: 10000 alpha: 0.020000
   1/    1 sec. ...........................................................
   0/    0 sec. ...........................................................
-76328547148648753048194563243184098580820083554903040130360678021668839666631610
325110251474922949751138950837715669121812685389824.000000: 0.000000
   SSS
   SSS
   SSSS
   SSS
    SS



This is a virus-infected file!
```

Fig. 16 SOM Umatrix and detection result of W95 Anxiety.1397 virus



```
./VirusDetector -din Win32.HLLP.Semisoft.b.exe.dat -cout Win32.HLLP.Semisoft.b.e
xe.cod
Training map, first part, rlen: 1000 alpha: 0.050000
   0/   0 sec. ...........................................................
Training map, second part, rlen: 10000 alpha: 0.020000
   0/   0 sec. ...........................................................
   1/   1 sec. ...........................................................
-11128292461603593674476817969515362957969416297689198789832905915443477629602004
10464219761696290596735822259896219482929785744453650622081967299021567747792174
01353004740657457224257310525107487866948026969458272208631501946880.000000: 0.0
00000
          $
   $      $
   $$
  $$$
   $


$
This is a virus-infected file!
```

Connected to duf-ssh.unifr.ch
SSH2 - aes128-cbc - hmac-md5 - none
81x20

Fig. 17 SOM Umatrix and detection result of Win32.HLLP.Semisoft virus

Table 1 Win9x virus detection result by *VirusDetector*

Virus name	Size	Date	Detection	Virus name	Size	Date	Detection
Altar.797	8,192	6-19-99	O	Altar.884	4,096	10-4-02	O
Altar.910	4,096	10-24-99	O	Antic.695	7,863	9-2-02	O
Anxiety.1358	1,52,778	3-1-04	O	Anxiety.1397	49,736	3-1-04	O
Anxiety.1399	8,192	3-21-98	O	Anxiety.1399.b	49,736	3-1-04	O
Anxiety.1422	5,684	11-22-03	O	Anxiety.1451	29,213	2-21-01	O
Anxiety.1486	8,192	1-24-98	O	Anxiety.1517	8,192	11-22-03	O
Anxiety.1586	42,590	11-22-03	O	Anxiety.1596	49,736	3-11-98	O
Anxiety.1823	8,192	3-1-04	O	Anxiety.1823.b	6,196	7-2-03	O
Anxiety.2471	8,192	11-22-03	O	Apop.1086	8,192	7-17-02	O
Argos.310	4,096	9-2-02	O	Argos.328	4,096	3-1-04	O
Argos.335	4,096	9-24-02	O	Argos.402	4,096	6-30-99	O
Bodgy.3230	97,438	4-15-03	X	Bonk.1232	19,632	11-22-03	O
Bonk.1243	9,460	3-1-04	O	Boza.2220	24,576	11-22-03	X
Boza.a	12,408	3-17-96	O	Boza.b	7,994	11-22-03	O
Boza.c	16,384	3-1-04	O	Boza.d	16,384	11-22-03	O
Boza.e	16,384	11-22-03	O	ByteSV.Thorn.886	20,480	11-22-03	O
Caw.1262	2,05,550	11-22-03	O	Caw.1335	5,943	3-1-04	O
Caw.1416	55,964	11-22-03	O	Caw.1419	54,667	11-22-03	X
Caw.1457	6,065	11-22-03	O	Caw.1458	8,192	10-5-02	O
Caw.1525	24,576	11-22-03	O	Caw.1531	8,192	11-2-01	O
Caw.1557	1,80,224	12-31-00	O	Chimera.1542	35,846	11-22-03	O
CIH	19,536	7-8-02	O	CIH.1003.b	4,608	3-1-04	O
CIH.1010.b	37,394	11-22-03	O	CIH.1016	34,304	9-2-02	O
CIH.1019.c	4,896	3-1-04	O	CIH.1024	1,553	3-1-04	O
CIH.1026	1,555	3-1-04	O	CIH.1031	4,096	9-2-02	O
CIH.1035	1,564	3-1-04	O	CIH.1040	1,59,744	3-1-04	O

Win9x Virus total number: 291, Error number: 26, False negative: 0.09 (approx. 9%)

Date indicates the virus's detected and caught date in form MM-DD-YY

Virus names are also the names of the test files. Size unit is byte

encrypted and 50 were polymorphic. This represents again approximately 16% of all tested Win32 virus-infected files. The way *VirusDetector* checks these files is identical to testing for unencrypted, non-polymorphic viruses. The results in the encrypted or polymorphic case are quite noticeable. In the case of Win9x executables, either encrypted or polymorphic viruses were detected easily by *VirusDetector* with 3% and 13% false negative rate, respectively. In case of Win32 executables containing an encrypted virus, the false negative rate was lower (13%) than *VirusDetector*'s average false negative rate on the entire data set (presented in Tables 4 and 5). However, the false negative rate for Win32 executables infected with a polymorphic virus was much higher (42%) than average (16%) (see Tables 6 and 7).

All encrypted Win9x viruses we used in our tests are listed in Table 4, all polymorphic Win9x viruses are in Table 6, all encrypted Win32 viruses are in Table 5, and all polymorphic Win32 viruses are in Table 7, each attached with their size, the date when they were caught and the test result of *VirusDetector*. For information about these polymorphic and encrypted parasitic

Table 2 Win32 Virus detection result by *VirusDetector*

Virus name	Size	Date	Detection	Virus name	Size	Date	Detection
Adson.1559	8,192	7-26-02	O	Adson.1734	20,480	11-22-03	O
Aidlot	8,192	10-4-04	O	Akez	32,768	4-27-02	O
Aliser.7825	12,288	3-6-03	O	Aliser.7897	8,192	6-8-03	O
Aliser.8381	8,192	6-8-03	O	Alma.2414	10,606	3-1-04	O
Awfull.2376	3,072	9-9-03	O	Awfull.3571	4,096	3-1-04	O
Bakaver.a	24,576	10-6-03	O	Banaw.2157	8,192	11-22-03	O
Barum.1536	5,632	8-31-02	O	Bee	24,576	3-1-04	O
Beef.2110	57,344	3-1-04	O	Belial.2537	8,192	11-22-03	O
Belial.2609	2,54,513	3-9-02	X	Belial.a	4,096	3-10-04	O
Belial.b	4,096	2-23-02	O	Belial.c	4,096	11-22-03	O
Belial.d	4,096	9-24-02	O	Belod.a	8,192	3-12-02	O
Belod.b	8,192	8-21-02	O	Belod.c	8,192	3-14-02	O
Bender.1363	3,584	12-31-01	O	Bika.1906	8,192	12-31-01	O
BingHe	2,96,643	10-11-02	X	Blackcat.2537	8,192	9-9-03	O
Blakan.2016	8,192	12-31-01	O	Blateroz	8,192	9-2-02	O
Blueballs.4117	16,384	11-22-03	X	Bobep	8,192	8-25-03	O
Bogus.4096	38,400	10-13-99	O	Bolzano.2122	36,864	2-10-03	O
Bolzano.2664	15,135	2-10-03	O	Bolzano.2676	15,183	2-10-03	O
Bolzano.2716	13,521	2-10-03	O	Bolzano.3100	15,277	2-10-03	O
Bolzano.3120	15,331	2-10-03	O	Bolzano.3148	15,373	2-10-03	O
Bolzano.3164	15,409	2-10-03	O	Bolzano.3192	15,457	3-1-04	O
Bolzano.3628	16,095	3-1-04	O	Bolzano.3904	16,251	2-10-03	O
Bolzano.5572	28,237	2-10-03	O	Butter	96,665	9-2-02	X

Win32 Virus total number: 499, Error number: 103, False negative: 0.2064 (approx. 21%)

Date indicates the virus's detected and caught date in form MM-DD-YY

Virus names are also the names of the test files. Size unit is Byte

Table 3 Normal executable program's virus check result by *VirusDetector*

Filename	Detection	Filename	Detection
dxwebsetup.exe		divx311.exe	
awsepersonal.exe		DVD2DIVXVCD_trial.exe	
paulp_en1.exe		adrenalin2.0.1.exe	
GOMPLAYER14.exe		sdvd190.exe	
csdl13.exe		HwpViewer.exe	
SSHSecureShellClient-3.2.9.exe		DivX505Bundle.exe	
klcodec220b.exe		SwansMP24a-WCP.exe	
DivXPro511GAINBundle.exe		NATEON.exe	X
WinPcap_3_1_beta_3.exe		ducp708_type3_free.exe	
wmpcdcs8.exe		Acrobat.exe	
iTunes.exe		pccmain.exe	
sicstusc.exe		Tra.exe	
acrodist.exe		java.exe	X
PCcpfw.exe		sicstus.exe	
Trialmsg.exe	X	Ad-Aware.exe	
javaw.exe	X	PCctool.exe	
splfr.exe	X	tsc.exe	
AdobeUpdateManage.exe		jpicp132.exe	X
Photoshp.exe		spmkds.exe	
conf.exe		policytool.exe	X
spmkrs.exe	X	unregaaw.exe	
CSDL.exe		kinit.exe	X
Sshclient.exe		ssh2.exe	

Normal executable file's total number: 80, Error number: 24, False positive: 0.3 (approx. 30%)

If the result of detection is marked X, *VirusDetector* says this file is a virus-infected file, which means incorrect detection

viruses, please refer to the site about Windows viruses at KASPERSKY (Metropolitan Network BBS Inc., Bern, Switzerland) [15].

3.3 False positives versus false negatives in *VirusDetector*

A false positive occurs when we test a non-infected file and the test result categorizes the file as positive (i.e. infected). In the inverse case of a not-detected-infected file, the outcome is called a false negative. The false negative rate and the false positive rate are interdependent; to decrease one is to increase the other. Therefore, it is important to decide which side to decrease and which to increase. A significant role is played by the factor value, one uses it in *VirusDetector* to decide whether the value of SOM cell is significant or not (i.e. whether or not the cell is likely to present a fraction of the virus

Table 4 Win9x encrypted parasitic virus detection result by *VirusDetector*

Virus name	Size	Date	Detection	Virus name	Size	Date	Detection
Bumble.1736	8,192	3-1-04	O	Bumble.1738	8,192	11-22-03	O
Iced.1344	44,032	9-6-01	O	Iced.1376	8,192	11-22-03	O
Iced.1412	8,192	11-22-03	O	Iced.1617	8,192	3-1-04	O
Iced.2112	8,192	3-1-04	O	Mad.2736.a	32,768	10-4-04	O
Mad.2736.b	32,768	10-4-04	O	Mad.2736.c	32,768	1-7-02	O
Mad.2736.d	32,768	1-7-02	O	Mad.2806	32,768	1-19-98	O
Nathan.3276	7,372	9-2-02	O	Nathan.3520.a	12,288	3-10-04	O
Nathan.3520.b	16,384	9-9-01	O	Nathan.3792	1,85,552	10-23-99	O
Obsolete.1419	5,003	3-1-04	O	PoshKill.1398	8,192	4-21-01	O
PoshKill.1406	8,192	3-17-03	O	PoshKill.1426	8,192	8-20-01	O
PoshKill.1445.a	8,192	3-10-04	O	PoshKill.1445.b	8,192	11-22-03	O
Priest.1419	4,096	8-9-99	O	Priest.1454	4,096	3-1-04	O
Priest.1478	9,728	6-10-00	X	Priest.1486	9,728	10-4-01	O
Priest.1495	9,728	8-18-01	O	Priest.1521	9,728	3-1-04	O
Shoerec	3,21,536	8-12-01	O	Tip.2475	10,752	11-22-03	O
Voodoo.1537	61,441	11-22-03	O	Werther.1224	6,344	12-31-01	O

Encrypted Win9x Virus total number: 30, Error number: 1, False negative: 0.03 (3%)

Table 5 Win32 encrypted parasitic virus detection result by *VirusDetector*

Virus name	Size	Date	Detection	Virus name	Size	Date	Detection
Ditto.1488	6,096	3-1-04	O	Ditto.1492	12,288	11-22-03	O
Ditto.1539	8,192	10-1-00	O	Gloria.2820	16,384	11-22-03	X
Gloria.2928	16,384	3-1-04	O	Gloria.2963	12,288	10-1-00	O
Idele.2104	8,192	7-8-03	O	Idele.2108	8,192	3-1-04	O
Idele.2160	8,192	11-12-03	O	IhSix.3048	8,192	11-22-03	O
Infinite.1661	8,192	3-1-04	O	Levi.2961	12,288	5-16-01	O
Levi.3040	7,188	11-22-03	O	Levi.3090	12,288	11-22-03	O
Levi.3137	35,941	11-22-03	O	Levi.3205	12,288	3-1-04	X
Levi.3240	16,384	8-17-02	O	Levi.3244	16,384	11-22-03	X
Levi.3432	16,384	11-22-03	O	Mix.1852	4,096	5-30-00	O
Niko.5178	65,611	11-22-03	X	Santana.1104	81,920	12-4-01	O
Savior.1680	8,192	1-8-01	O	Savior.1696	12,288	5-18-01	O
Savior.1740	12,288	3-28-02	O	Savior.1828	20,480	8-6-01	O
Savior.1832	12,288	3-1-04	O	Savior.1904	12,288	12-4-01	O
Undertaker.4887	12,288	11-22-03	O	Undertaker.5036.a	12,288	11-22-03	O

Encrypted Win32 Virus total number: 30, Error number: 4, False negative: 0.13 (13%)

Table 6 Win9x Polymorphic virus detection result by *VirusDetector*

Virus name	Size	Date	Detection	Virus name	Size	Date	Detection
Begemot	8,192	3-1-04	O	Darkmil.5086	12,288	3-1-04	X
Darkmil.5090	74,210	7-2-03	O	Fiasko.2500.a	8,192	11-22-03	O
Fiasko.2500.b	12,935	3-1-04	O	Fiasko.2508	8,192	3-1-04	O
Invir.7051	9,728	3-1-04	O	Luna.2636	8,192	4-24-02	O
Luna.2757.a	62,213	3-10-04	O	Luna.2757.b	12,288	1-1-80	O
Marburg.a	4,93,789	3-10-04	O	Marburg.b	28,381	11-22-03	X
Matrix.3597	35,916	2-14-03	O	Merinos.1763	9,216	3-1-04	O
Merinos.1849	8,192	11-22-03	O				

Win9x Polymorphic Virus total number: 15, Error number: 2, False negative: 0.13 (13%)

mask). To determine the threshold of decision in certain patterns, the test results of all 790 virus-infected and 80 non-infected Windows executable files were taken into account. For different factor values, the false-positive and false-negative rates of the tests on the entire data set are presented in Figure 18.

As Figure 18 shows, the false positives remain more or less in the same range (0.25–0.3) for factor values 72 and 79. On the other hand, the false-negative rate is increased significantly. Thus, a factor value of 72 is selected in *VirusDetector* to achieve a false-negative rate of approximately 16% and keep the false-positive rate below 30%. So, without knowing any virus signature and anything else about a virus, we can detect a virus infection with a probability of 84% and false positive rate of 30%. Some further fine-tuning might be necessary, possibly on the cost of reducing the detection capabilities (i.e. increasing the false-negative rate), in order to decrease the false positives.

Using the factor value (72), all the files (790 virus-infected files and 80 non-virus-infected normal executable files) were tested. The full list of information

Table 7 Win32 polymorphic virus detection result by *VirusDetector*

Virus name	Size	Date	Detection	Virus name	Size	Date	Detection
Andras.7300	14,238	7-27-02	O	AOC.2044	8,192	11-28-99	O
AOC.2045	8,192	11-26-99	O	AOC.3657	16,384	3-1-04	O
AOC.3833	16,384	3-1-04	O	AOC.3860	20,480	2-10-03	X
AOC.3864	20,480	2-10-03	O	Champ	12,288	2-10-03	O
Champ.5430	12,288	10-6-02	O	Champ.5464	12,288	2-10-03	O
Champ.5477	12,288	10-6-02	O	Champ.5495	6,144	2-10-03	X
Champ.5521	12,288	2-10-03	X	Champ.5536	12,288	2-10-03	X
Champ.5714	12,288	2-10-03	O	Champ.5722	16,384	2-10-03	X
Chop.3808	64,049	8-29-01	X	Crypto	49,152	3-1-04	X
Crypto.a	28,672	11-22-03	X	Crypto.b	32,768	11-22-03	O
Crypto.c	32,768	11-22-03	O	Driller	94,208	3-1-04	O
Harrier	1,08,544	11-22-03	X	Hatred.a	16,384	3-10-04	O
Hatred.d	16,384	10-29-02	O	Kriz.3660	4,15,232	7-27-02	X
Kriz.3740	7,64,928	10-4-04	X	Kriz.3863	4,75,136	10-4-04	X
Kriz.4029	12,288	3-1-04	O	Kriz.4037	12,288	8-19-01	O
Kriz.4050	4,79,232	11-22-03	X	Kriz.4057	12,288	8-19-01	X
Kriz.4075	12,288	11-22-03	O	Kriz.4099	12,288	11-22-03	X
Kriz.4233	8,192	7-15-01	X	Kriz.4271	57,344	10-4-04	O
Prizm.4428	8,704	9-4-02	X	RainSong.3874	8,192	11-22-03	O
RainSong.3891	61,509	3-1-04	O	RainSong.3910	8,192	12-5-01	X
RainSong.3956	12,288	10-4-04	X	RainSong.4198	8,192	3-12-02	O
RainSong.4266	12,288	10-4-04	X	Thorin.11932	16,384	3-1-04	O
Thorin.b	16,384	3-1-04	O	Thorin.c	16,384	10-23-99	O
Thorin.d	16,384	7-14-99	O	Thorin.e	16,384	10-23-99	O
Vampiro.7018	18,432	3-1-04	X	Vampiro.a	16,896	3-10-04	O

Encrypted Win32 Virus total number: 50, Error number: 21, False negative: 0.42 (42%)

and results on the tested files is given in the appendix. Among the 291 Win9x virus-infected files, *VirusDetector* failed to detect 26 (approximately 9%, see Table 1). Among the 499 Win32 virus-infected files, *VirusDetector* did not succeed in detecting 103 (approximately 21%, see Table 2). On the other hand, among the 80 non-infected Windows executable files, *VirusDetector* failed to pass 24 (approximately 30%; see Table 3).

4 Discussion and conclusions

The virus-detection approach presented in this paper exploits the detection capabilities of a SOM. It basically uses structural information about the data contained in an executable file: the virus code is data injected into a formerly complete and (sort of) homogeneous structure, namely the program code. Hence the virus, even though not easily detectable by standard techniques (assuming that the virus signature is not known), is “somewhat different” from the program it infected. The SOM in the non-standard way we used it, is capable of doing just that: reflecting the presence of data in an executable file which is somehow different from the rest. Whether the injected code was an encrypted parasitic or a polymorphic parasitic virus does not matter; it still differs from the rest of the program code. In SOM terminology: its data density, compared to the original program code, is high enough to display what we call a virus mask in this paper (i.e. an area of close neighborhood in the SOM projection).

Standard anti-virus software can detect variants of a virus only if different virus signatures are available. However, the detection approach in this paper detects viruses independent of any prior knowledge (such as a virus mask). Even polymorphic or encrypted parasitic viruses show a virus mask which can be detected by *VirusDetector*. Since the question of whether or not a program contains virus code is, in general, undecidable, *VirusDetector* cannot be perfect. It is therefore a good result that in our tests (with 790 files, all infected by a different virus or different version of a virus), *VirusDetector* detected almost 84% of the viruses with a false positive rate of 30%. During our experiments, based on the good detection rate of *VirusDetector*, our confidence in the possibility of detecting unknown viruses using our system increased significantly.

Until now, the classical virus-detection techniques could not deal with unknown viruses (not all, but some). The SOM-based, non-signature-based virus detection complements these standard techniques in that it provides a tool capable of identifying unknown viruses. The combination of signature-based methods with our approach can make systems much more secure by making them less vulnerable to infections by unknown viruses.

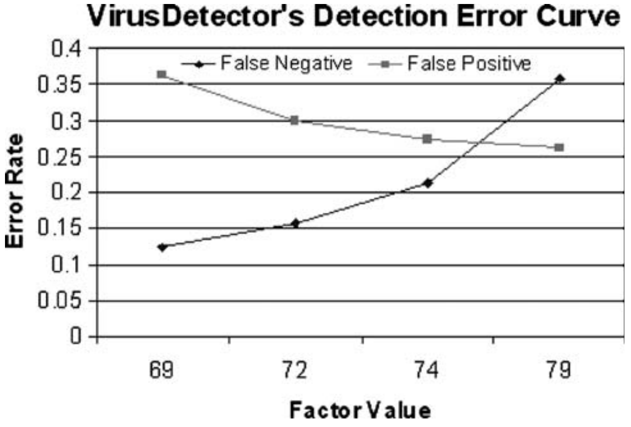


Fig. 18 *VirusDetector*'s error curve based on factor values

There is still the macro-virus-detection problem. Although the SOM visualization pattern looks very similar to a virus mask in a NewEXE file, it will always be produced when a document is checked for macro viruses using *VirusDetector*. An approach different from that for parasitic viruses is therefore needed to deal with macro viruses. Even though the macro virus is inserted into a complete document, it is first of all part of a macro and that macro is then saved in the macro area of a document file. Because of this, the inserted macro virus data can not be differentiated from the macro itself and identified by the SOM neurons. Additionally, the macro virus part is not big enough to produce a significant neighborhood density and is possibly “hidden behind” low density data. The macro part is simply too small compared to the entire data structure. Future work will aim to overcome this problem and be able to deal with macro viruses as well.

Another remaining weakness is the too high false-positive rate of 30%. Even though the experimental results are very promising and to some extent even surprising, additional work must be spent in the future on reducing the relative number of false positives. There is already some scope for that in our approach by adapting the factor value used by *VirusDetector*. This will cause the false-negative rate to increase, which is partly acceptable. However, we will also examine other improvements in order to reduce the relative number of false positives. Still, even if the false-negative rate must be increased for the sake of reducing the relative number of false positives, if *VirusDetector* can only prevent a single unknown virus from infecting our system (or systems world-wide), the significant research effort spent on developing *VirusDetector* was absolutely worth it.

Acknowledgements We would like to thank VX Heavens! “(http://vx. netlux.org/)”. This site is dedicated to providing information about computer viruses (or virii, as some would prefer) to anyone who is interested in this topic. Almost all virus test samples used for virus detection in this paper are available in this site.

5 Appendix

All the test results of *VirusDetector* are listed here. Tested files were either Win9x or Win32 executable files (Tables 8, 9 and 10)

Table 8 Win9x virus detection result by *VirusDetector*

Virus name	Size	Date	Detection	Virus name	Size	Date	Detection
Altar.797	8,192	6-19-99	O	Altar.884	4,096	10-4-02	O
Altar.910	4,096	10-24-99	O	Antic.695	7,863	9-2-02	O
Anxiety.1358	152,778	3-1-04	O	Anxiety.1397	49,736	3-1-04	O
Anxiety.1399	8,192	3-21-98	O	Anxiety.1399.b	49,736	3-1-04	O
Anxiety.1422	5,684	11-22-03	O	Anxiety.1451	29,213	2-21-01	O
Anxiety.1486	8,192	1-24-98	O	Anxiety.1517	8,192	11-22-03	O
Anxiety.1586	42,590	11-22-03	O	Anxiety.1596	49,736	3-11-98	O
Anxiety.1823	8,192	3-1-04	O	Anxiety.1823.b	6,196	7-2-03	O
Anxiety.2471	8,192	11-22-03	O	Apop.1086	8,192	7-17-02	O
Argos.310	4,096	9-2-02	O	Argos.328	4,096	3-1-04	O
Argos.335	4,096	9-24-02	O	Argos.402	4,096	6-30-99	O
Arianne.1022.a	4,606	3-10-04	O	Arianne.1022.b	94,112	11-22-03	O
Arianne.1052	8,192	5-16-02	O	Atom.4790	64,182	5-8-02	X
Babylonia.11036	33,734	3-1-04	X	Babylonia.attach	5,984	5-16-02	O
Babylonia.Plugin.Dropper	12,606	11-22-03	X	Babylonia.Plugin.Greetz	621	11-22-03	X
Babylonia.Plugin.IrcWorm	1,707	11-22-03	O	Babylonia.Plugin.Poll	1,041	11-22-03	O
Begemot	8,192	3-1-04	O	BlackBat.2615	8,192	5-31-01	O
BlackBat.2787	8,192	11-22-03	O	BlackBat.2795	8,192	11-22-03	O
BlackBat.2840	8,192	5-27-01	X	BlackBat.2841.a	8,192	3-10-04	O
BlackBat.2841.b	8,192	5-31-01	X	BlackBat.2988	8,192	11-22-03	O

Table 8 continued

Virus name	Size	Date	Detection	Virus name	Size	Date	Detection
Bodgy.3230	97,438	4-15-03	X	Bonk.1232	19,632	11-22-03	O
Bonk.1243	9,460	3-1-04	O	Boza.2220	24,576	11-22-03	X
Boza.A	12,408	9-2-99	O	Boza.C	16,384	9-10-99	O
Boza.a	12,408	3-17-96	O	Boza.b	7,994	11-22-03	O
Boza.c	16,384	3-1-04	O	Boza.d	16,384	11-22-03	O
Boza.e	16,384	11-22-03	O	Bumble.1736	8,192	3-1-04	O
Bumble.1738	8,192	11-22-03	O	Butool.910	14,222	9-2-02	O
Buzum.1828	6,310	3-1-04	O	ByteSV.Thorn.886	20,480	11-22-03	O
Caw.1262	2,05,550	11-22-03	O	Caw.1335	5,943	3-1-04	O
Caw.1416	55,964	11-22-03	O	Caw.1419	54,667	11-22-03	X
Caw.1457	6,065	11-22-03	O	Caw.1458	8,192	10-5-02	O
Caw.1525	24,576	11-22-03	O	Caw.1531	8,192	11-2-01	O
Caw.1557	1,80,224	12-31-00	O	Chimera.1542	35,846	11-22-03	O
CIH	19,536	7-8-02	O	CIH.v1.2	19,536	9-12-99	O
CIH.v1.3	36,864	9-3-99	O	CIH.v1.4	4,608	9-22-99	O
CIH.1003.b	4,608	3-1-04	O	CIH.1010.b	37,394	11-22-03	O
CIH.1016	34,304	9-2-02	O	CIH.1019.c	4,896	3-1-04	O
CIH.1024	1,553	3-1-04	O	CIH.1026	1,555	3-1-04	O
CIH.1031	4,096	9-2-02	O	CIH.1035	1,564	3-1-04	O
CIH.1040	1,59,744	3-1-04	O	CIH.1042	53,248	3-1-04	O
CIH.1048	20,480	9-2-02	O	CIH.1049	65,536	9-2-02	O
CIH.1103	2,144	11-22-03	O	CIH.1106	1,53,088	11-27-02	O
CIH.1122	1,651	9-2-02	O	CIH.1129	1,658	3-1-04	X
CIH.1142	16,384	6-19-98	O	CIH.1230	1,759	3-1-04	O
CIH.1262	1,778	3-1-04	O	CIH.1297	1,826	11-22-03	O
CIH.1363	59,392	3-1-04	X	CIH.2563	24,576	9-2-02	X
CIH.2690	94,208	9-2-02	X	CIH.816.a	59,392	9-2-02	X
CIH.816.b	5,120	9-2-02	O	CIH.862	1,903	5-10-02	O
CIH.876	20,480	9-2-02	X	CIH.913	20,480	9-2-02	O
CIH.937	20,480	9-2-02	O	CIH.973	1,502	3-1-04	O
CIH.corrupted	1,88,478	1-10-02	O	CIH.dam	1,090,318	1-1-03	X
CIH-II.776	53,248	9-2-02	O	CIH-II.882	20,480	9-29-02	O
CIH.intended	2,426	8-31-99	O	CIH-Killer.1373	9,053	3-1-04	O
CIH.src	3,01,056	3-1-04	X	Companion.4096	4,096	3-1-04	O
Croman	16,384	9-2-02	X	Dado	3,32,817	3-1-04	X
Darkmil.5086	12,288	3-1-04	X	Darkmil.5090	74,210	7-2-03	O
DarkSide.1105	8,192	4-6-02	O	DarkSide.1371	8,192	3-1-04	O
DarkSide.1491	9,728	10-4-04	O	Dead.1086	8,192	9-29-02	O
Dead.4172	11,392	9-2-02	O	Dead.4316	16,796	5-16-02	O
Dead.4388	11,608	9-2-02	O	Demo.8192	8,192	11-22-03	O
Dodo.1022	8,192	9-2-02	O	Dupator.1503	1,10,592	1-29-04	O
Eak	1,03,424	12-19-02	X	Esmeralda.807	4,955	7-15-01	O
Etymo.1308	7,168	3-1-04	O	Evil.953.a	60,345	11-22-03	X
Evil.953.b	4,096	9-24-02	O	Evil.962	8,192	3-1-04	O
Evil.962.b	35,266	11-22-03	O	Evil.962.c	4,096	8-26-01	O
Federal	8,192	9-2-02	O	Fiasko.2500.a	8,192	11-22-03	O
Fiasko.2500.b	12,935	3-1-04	O	Fiasko.2508	8,192	3-1-04	O
Filth.1030	4,096	3-1-04	O	Flee.835	8,192	11-22-03	O
Fono.15327	24,064	3-1-04	X	Fono.Trojan	263	5-16-02	O
Frone.864	69,632	5-16-02	X	Frone.951	8,192	9-2-02	O
FYS.1728	8,192	7-15-01	O	Gara.640	8,192	11-22-03	O
Gara.842.a	8,192	3-10-04	O	Gara.842.b	8,192	11-22-03	O
Gara.917	8,192	12-4-01	O	Harry.a	8,192	3-10-04	O
Harry.b	8,192	6-2-97	O	Hooy.8192	32,768	3-28-00	O
Horn.1851	6,322	3-1-04	O	Horn.1862	6,334	11-22-03	O
Horn.2223	6,695	11-22-03	O	Horn.2245	6,719	3-1-04	O
HPS.5124	26,563	3-1-04	X	I13.a	8,192	11-22-03	O
I13.b	12,288	3-1-04	O	I13.c	8,192	3-1-04	O
I13.d	12,288	11-22-03	X	I13.e	8,192	11-22-03	O
I13.f	8,192	10-4-02	O	Iced.1344	44,032	9-6-01	O
Iced.1376	8,192	11-22-03	O	Iced.1412	8,192	11-22-03	O

Table 8 continued

Virus name	Size	Date	Detection	Virus name	Size	Date	Detection
Iced.1617	8,192	3-1-04	O	Iced.2112	8,192	3-1-04	O
Icer.541	4,096	11-22-03	O	Icer.619	1,92,512	1-13-04	O
ILMX.1291	53,248	3-1-04	O	Invir.7051	9,728	3-1-04	O
Jacky.1440	4,646	3-1-04	O	Jacky.1443	8,192	11-22-03	O
Javel.512	1,529	3-1-04	O	Julus.1904.a	8,192	11-22-03	O
Julus.1904.b	8,192	10-4-04	O	Julus.1929.a	8,192	11-22-03	O
Julus.1929.b	8,192	1-23-03	O	Julus.2702.a	12,288	11-22-03	O
Julus.2702.b	8,192	8-29-01	O	Julus.2777	12,288	11-22-03	O
K32.1012	5,108	11-22-03	O	K32.2929	8,192	10-4-02	O
K32.3030	9,174	3-1-04	O	K32.Roma.2929	9,643	12-9-00	O
Kaze	20,480	8-8-02	O	Kurgan.10240	14,336	9-2-02	O
Lizard.1967	7,099	3-1-04	O	Lizard.2381	2,957	3-1-04	O
Lizard.2869	3,715	3-1-04	O	Lizard.5150	5,150	10-4-04	O
Lorez.1766.a	8,192	3-10-04	O	Lorez.1766.b	8,192	6-28-01	O
LoveSong.998	61,440	12-4-01	O	Lud.Hill.401	8,192	3-1-04	O
Lud.Jadis.3567	9,216	3-1-04	O	Lud.Jadis.3579	9,216	11-22-03	O
Lud.Jez.676	8,192	3-1-04	O	Lud.Jez.682	8,192	11-22-03	O
Lud.Yel.1886	22,141	10-4-04	O	Luna.2636	8,192	4-24-02	O
Luna.2757.a	62,213	3-10-04	O	Luna.2757.b	12,288	1-1-80	O
Mad.2736.a	32,768	10-4-04	O	Mad.2736.b	32,768	10-4-04	O
Mad.2736.c	32,768	1-7-02	O	Mad.2736.d	32,768	1-7-02	O
Mad.2806	32,768	1-19-98	O	Marburg.a	493,789	3-10-04	O
Marburg.b	28,381	11-22-03	X	MarkJ.826	8,192	3-1-04	O
Matrix.3597	35,916	2-14-03	O	Memorial	35,515	2-7-98	O
Merinos.1763	9,216	3-1-04	O	Merinos.1849	8,192	11-22-03	O
MMort.1335	8,192	11-22-03	O	MMort.1340	8,192	11-22-03	O
MMort.1348	8,192	3-1-04	O	MMort.1366	8,192	10-23-99	O
Molly.680	8,192	5-16-02	O	Molly.722	8,192	3-1-04	O
MrKlunky.a	6,943	3-10-04	O	MrKlunky.b	6,779	7-2-03	X
MSpawn.4608	8,897	11-22-03	O	Murkry.383	4,096	5-15-02	O
Murkry.398.a	59,392	11-22-03	O	Murkry.398.b	4,096	9-24-02	O
Murkry.399	4,096	3-1-04	O	Murkry.441	26,624	9-24-02	O
Nathan.3276	7,372	9-2-02	O	Nathan.3520.a	12,288	3-10-04	O
Nathan.3520.b	16,384	9-9-01	O	Nathan.3792	1,85,552	10-23-99	O
Noise.414	57,344	3-1-04	O	Obsolete.1419	5,003	3-1-04	O
Onerin.371	4,096	1-9-02	O	Onerin.383	4,096	1-9-02	O
Opa.1103	45,056	5-8-01	O	Opa.1149	45,056	7-15-01	O
Padania.1335	8,192	3-1-04	O	Paik.1908	8,192	5-14-01	O
PoshKill.1398	8,192	4-21-01	O	PoshKill.1406	8,192	3-17-03	O
PoshKill.1426	8,192	8-20-01	O	PoshKill.1445.a	8,192	3-10-04	O
PoshKill.1445.b	8,192	11-22-03	O	Powerful.1592	6,144	3-1-04	X
Powerful.1773	6,144	3-1-04	O	Powerful.1901	12,288	7-12-01	O
Priest.1419	4,096	8-9-99	O	Priest.1454	4,096	3-1-04	O
Priest.1478	9,728	6-10-00	X	Priest.1486	9,728	10-4-01	O
Priest.1495	9,728	8-18-01	O	Priest.1521	9,728	3-1-04	O
Prizm.4428	8,704	9-4-02	X	Puma.1024	4,096	3-1-04	O
Regix.4096.a	8,192	3-10-04	O	Sanat.3151	16,384	3-1-04	X
Shoerec	321,536	8-12-01	O	Sign.2028	8,192	1-8-01	O
Smash.10262	16,384	8-16-01	X	SST.952	4,096	3-1-04	O
Tecata.1761	66,029	10-4-04	O	Tenrobot.b	49,152	5-14-03	X
Tip.2475	10,752	11-22-03	O	Titanic.3214	7,822	8-16-01	O
Uwaga.3237	8,192	11-22-03	O	Vivic	8,192	8-17-02	O
Voodoo.1537	61,441	11-22-03	O	Werther.1224	6,344	12-31-01	O
Whal.a	8,192	1-24-01	O	Whyg.1193	8,192	6-24-01	O
Yabran.3132	4,608	3-1-04	O	Yobe	20,480	3-1-04	O
Youd.1388	8,192	11-22-03	O	Yoyo.653	4,096	3-1-04	O
Zerg.3849	8,192	3-1-04	O	Zofo.848	20,480	3-1-04	O
Zoual	147,456	10-18-02	X				

Win9x Virus total number: 291, Error number: 26, False negative: 0.09 (approx. 9%)

Date indicates the virus's detected and caught date in form MM-DD-YY

Virus names are also the names of the test files. Size unit is byte

Table 9 Win32 virus detection result by *VirusDetector*

Virus name	Size	Date	Detection	Virus name	Size	Date	Detection
Adson.1559	8,192	7-26-02	O	Adson.1734	20,480	11-22-03	O
Aidlot	8,192	10-4-04	O	Akez	32,768	4-27-02	O
Aliser.7,825	12,288	3-6-03	O	Aliser.7,897	8,192	6-8-03	O
Aliser.8,381	8192	6-8-03	O	Alma.2414	10,606	3-1-04	O
Alma.37195	45,387	11-22-03	X	Alma.37274	40,960	4-19-02	X
Alma.5319	13,511	3-1-04	X	Andras.7300	14,238	7-27-02	O
AOC.2044	8,192	11-28-99	O	AOC.2045	8,192	11-26-99	O
AOC.3657	16,384	3-1-04	O	AOC.3833	16,384	3-1-04	O
AOC.3860	20,480	2-10-03	X	AOC.3864	20,480	2-10-03	O
Apathy.5378	8,192	3-1-04	O	Apparition	96,239	12-22-99	O
Apparition.a	5,42,861	3-10-04	X	Apparition.b	1,67,707	6-5-98	X
Arianne.1052	6,684	3-11-02	O	Aris	3,31,785	3-1-04	X
Arrow.a	2,048	10-5-04	O	Artelad.2173	23,040	12-31-00	O
Asorl.a	32,269	3-10-04	X	AutoWorm.3072	3,072	3-1-04	O
Awfull.2376	3,072	9-9-03	O	Awfull.3571	4,096	3-1-04	O
Bakaver.a	24,576	10-6-03	O	Banaw.2157	8,192	11-22-03	O
Barum.1536	5,632	8-31-02	O	Bee	24,576	3-1-04	O
Beef.2110	57,344	3-1-04	O	Belial.2537	8,192	11-22-03	O
Belial.2609	2,54,513	3-9-02	X	Belial.a	4,096	3-10-04	O
Belial.b	4,096	2-23-02	O	Belial.c	4,096	11-22-03	O
Belial.d	4,096	9-24-02	O	Belod.a	8,192	3-12-02	O
Belod.b	8,192	8-21-02	O	Belod.c	8,192	3-14-02	O
Bender.1363	3,584	12-31-01	O	Bika.1906	8,192	12-31-01	O
BingHe	2,96,643	10-11-02	X	Blackcat.2537	8,192	9-9-03	O
Blakan.2016	8,192	12-31-01	O	Blateroz	8,192	9-2-02	O
Blueballs.4117	16,384	11-22-03	X	Bobep	8,192	8-25-03	O
Bogus.4096	38,400	10-13-99	O	Bolzano.2122	36,864	2-10-03	O
Bolzano.2664	15,135	2-10-03	O	Bolzano.2676	15,183	2-10-03	O
Bolzano.2716	13,521	2-10-03	O	Bolzano.3100	15,277	2-10-03	O
Bolzano.3120	15,331	2-10-03	O	Bolzano.3148	15,373	2-10-03	O
Bolzano.3164	15,409	2-10-03	O	Bolzano.3192	15,457	3-1-04	O
Bolzano.3628	16,095	3-1-04	O	Bolzano.3904	16,251	2-10-03	O
Bolzano.5572	28,237	2-10-03	O	Butter	96,665	9-2-02	X
Cabanas.a	7,171	5-7-04	X	Cabanas.b	7,171	3-1-04	O
Cabanas.Debug	95,748	10-4-04	O	Cabanas.e	16,384	7-8-03	O
Cabanas.MsgBox	39,996	10-4-04	X	Cabanas.Release	49,152	1-26-99	O
CabInfector	4,096	3-1-04	O	Cecile	28,672	12-31-01	X
Cefet.3157	7,253	9-2-02	O	Cerebrus.1482	8,192	3-1-04	O
Champ	12,288	2-10-03	O	Champ.5430	12,288	10-6-02	O
Champ.5464	12,288	2-10-03	O	Champ.5477	12,288	10-6-02	O
Champ.5495	6,144	2-10-03	X	Champ.5521	12,288	2-10-03	X
Champ.5536	12,288	2-10-03	X	Champ.5714	12,288	2-10-03	O
Champ.5722	16,384	2-10-03	X	Chatter	22,528	1-13-03	O
Chop.3808	64,049	8-29-01	X	Cornad	4,096	6-22-03	O
Crosser	1,02,400	12-6-03	X	Crypto	49,152	3-1-04	X
Crypto.a	28,672	11-22-03	X	Crypto.b	32,768	11-22-03	O
Crypto.c	32,768	11-22-03	O	Damm.1624	24,576	3-1-04	O
Damm.1628	4,096	5-16-02	O	Damm.1647.a	12,288	11-22-03	O
Datus	1,05,472	5-16-02	X	Delikon	16,384	1-6-04	O
Devir	24,576	10-4-04	O	Dictator.2304	10,496	11-22-03	X
Dislex	1,35,239	3-1-04	X	Ditex	2,12,992	4-17-02	O
Ditto.1488	6,096	3-1-04	O	Ditto.1492	12,288	11-22-03	O
Ditto.1539	8,192	10-1-00	O	Donny.a	8,192	3-10-04	O
Donut	12,800	3-1-04	O	Dream.4916	69,632	3-1-04	O
Driller	94,208	3-1-04	O	Drivalon.1876	3,072	7-8-03	O
Dudra.5632	12,288	7-7-01	O	Eclipse.a	8,192	3-10-04	O
Eclipse.b	6,644	3-15-01	O	Eclipse.c	8,192	8-18-99	O
Egolet.a	4,096	3-10-04	O	Egolet.b	4,096	7-7-02	O
Elerad	8,192	2-2-02	X	Emotion.a	4,608	3-10-04	O
Emotion.b	8,192	11-30-00	O	Emotion.c	8,192	2-10-03	O
Emotion.d	8,192	2-10-03	O	Emotion.gen	8,192	9-20-01	O

Table 9 continued

Virus name	Size	Date	Detection	Virus name	Size	Date	Detection
Enar	89,088	3-1-04	O	Enumiacs.6656	6,656	11-22-03	O
Enumiacs.8192.a	274	10-4-04	O	Fighter.a	6,656	11-22-03	O
Fighter.b	8,192	1-29-04	X	Flechal	69,632	3-1-04	O
Fosforo.a	8,192	3-10-04	O	Fosforo.b	8,192	10-24-02	O
Fosforo.c	8,192	12-31-01	X	Fosforo.d	8,192	3-5-03	X
Freebid	14,066	8-27-02	O	FunLove.4070	69,635	3-7-04	O
Gaybar	55,493	2-9-04	O	gen	8,192	9-3-02	O
Genu.a	8,192	11-22-03	O	Genu.b	8,192	11-22-03	O
Genu.c	5,619	8-18-02	O	Genu.d	8,192	7-26-02	O
Ghost.1667	8,192	3-1-04	O	Ginra.3334	8,966	8-31-02	O
Ginra.3413	8,192	9-2-02	O	Ginra.3570	8,192	5-18-02	O
Ginra.3657	8,192	10-4-04	O	Ginseng	4,096	8-24-02	O
Giri.4919	1,85,143	11-22-03	O	Giri.4970	13,162	3-9-02	O
Giri.5209	12,288	3-28-01	X	Gloria.2820	16,384	11-22-03	X
Gloria.2928	16,384	3-1-04	O	Gloria.2963	12,288	10-1-00	O
Glyn	8,192	3-1-04	O	Gobi.a	4,096	10-23-02	O
Godog	12,288	3-1-04	O	Golsys.14292	55,252	8-30-02	X
Grenp.2804	4,608	11-22-03	O	Halen.2593	8,192	3-1-04	O
Halen.2618	22,743	7-27-02	O	Halen.2619	8,192	7-12-02	O
Haless.1127	31,744	10-4-04	X	Harrier	1,08,544	11-22-03	X
Hatred.a	16,384	3-10-04	O	Hatred.d	16,384	10-29-02	O
Hawey	5,595	7-12-03	O	Heretic.1986	8,192	3-1-04	O
Hezhi	152,064	9-2-02	O	Hidrag.a	36,352	8-24-01	X
Highway.a	8,192	11-22-03	O	Highway.b	48,177	3-9-02	O
HIV	175	2-10-03	X	HIV.6340	12,288	11-22-03	O
HIV.6382	12,288	11-22-03	O	HIV.6386	12,288	10-4-04	X
HIV.6680	12,288	3-1-04	X	HLL.Fugo	55,808	7-6-04	X
HLLP.BadBy	3,29,728	11-22-03	X	HLLP.Bora.11264	11,264	5-8-02	X
HLLP.Clay	60,416	11-22-03	O	HLLP.Delvi	46,080	3-11-02	X
HLLP.Famer	22,528	1-18-03	X	HLLP.Freefall	36,352	6-15-02	X
HLLP.Gezad	28,672	7-8-03	O	HLLP.Givin	34,820	10-4-04	O
HLLP.Gosus	69,590	8-17-02	O	HLLP.Gotem	22,016	11-25-02	X
HLLP.Hetis	38,400	2-27-02	O	HLLP.Imel	34,816	8-30-02	O
HLLP.Karabah	1,90,988	11-22-03	O	HLLP.Kiro	79,632	4-9-04	X
HLLP.Lassa.40960	40,960	5-8-02	O	HLLP.Mincer	1,73,315	4-19-02	X
HLLP.MTV	68,096	11-22-03	X	HLLP.Nilob	24,576	7-14-00	O
HLLP.Pres	1,24,936	9-2-02	X	HLLP.Semisoft	59,904	12-4-99	O
HLLP.Shodi.c	98,318	4-9-04	X	HLLP.Sloc	1,04,448	8-31-02	O
HLLP.Sneak	34,816	3-1-04	O	HLLP.Thembe	130,322	3-1-04	X
HLLP.Unzi	24,576	3-25-02	O	HLLP.Winfig	33,280	11-22-03	O
HLLP.Yai	3,41,211	11-14-99	X	Htrip.a	8,192	12-4-01	O
Htrip.b	8,192	11-22-03	O	Htrip.c	8,192	12-4-01	O
Idele.2104	8,192	7-8-03	O	Idele.2108	8,192	3-1-04	O
Idele.2160	8,192	11-12-03	O	IhSix.3048	8,192	11-22-03	O
IKX	4,096	3-1-04	O	Infinite.1661	8,192	3-1-04	O
Infis.4608	4,608	3-1-04	O	Initx	2,10,432	10-4-04	X
Insom.1972.a	5,120	4-23-04	O	InvictusDLL.099	4,096	11-22-03	O
InvictusDLL.102	8,704	9-15-01	X	InvictusDLL.a	8,192	3-10-04	X
InvictusDLL.b	8,192	8-17-01	X	InvictusDLL.c	8,704	9-10-01	X
InvictusDLL.d	56,466	5-5-99	X	Ipamor.a	65,536	3-10-04	X
Ipamor.c	38,913	5-15-03	X	Ipamor.d	35,840	5-15-03	X
Ivaz	4,096	11-22-03	O	Jater	4,096	3-1-04	O
Jethro.5657	17,433	3-1-04	O	Junkcomp	65,536	12-30-02	X
Kala.7620	65,536	12-30-01	X	Kanban.a	3,072	3-10-04	O
Keisan.a	8,192	6-2-03	O	Keisan.b	8,192	6-2-03	O
Keisan.c	8,192	6-2-03	O	Keisan.d	8,192	6-2-03	O
Keisan.e	8,192	6-2-03	O	Ketan	4,096	3-1-04	O
Kiltex	1,55,648	3-1-04	O	Klinge	8,192	3-1-04	O
KME	36,864	3-1-04	O	KMKY	24,576	8-6-01	X
Knight.2350	6,958	12-4-01	O	Koru	65,536	5-15-98	O
Kriz.3660	4,15,232	7-27-02	X	Kriz.3740	7,64,928	10-4-04	X

Table 9 continued

Virus name	Size	Date	Detection	Virus name	Size	Date	Detection
Kriz.3863	4,75,136	10-4-04	X	Kriz.4029	12,288	3-1-04	O
Kriz.4037	12,288	8-19-01	O	Kriz.4050	4,79,232	11-22-03	X
Kriz.4057	12,288	8-19-01	X	Kriz.4075	12,288	11-22-03	O
Kriz.4099	12,288	11-22-03	X	Kriz.4233	8,192	7-15-01	X
Kriz.4271	57,344	10-4-04	O	Kuto.2058	10,250	8-31-02	X
Lad.1916	61,440	9-2-02	X	Ladmar.2004	57,344	9-1-02	O
Lamebyte	8,192	4-30-03	O	Lames.4096	8,192	1-3-98	O
Lamewin.1751	3,584	8-3-02	O	Lamewin.1813	3,584	4-15-02	O
Lamzan	8,192	5-18-03	O	Lanky.3153	12,288	5-1-02	O
LazyMin.31	96,768	4-9-04	X	Legacy	19,968	3-1-04	O
Levi.2961	12,288	5-16-01	O	Levi.3040	7,188	11-22-03	O
Levi.3090	12,288	11-22-03	O	Levi.3137	35,941	11-22-03	O
Levi.3205	12,288	3-1-04	X	Levi.3240	16,384	8-17-02	O
Levi.3244	16,384	11-22-03	X	Levi.3432	16,384	11-22-03	O
Lom	3,41,504	11-22-03	O	Lykov.a	9,338	6-9-03	X
Magic.1590	8,264	3-1-04	O	Magic.1922	8,192	9-2-02	O
Magic.3038	12,288	11-22-03	X	Magic.3078	12,288	8-17-02	O
Magic.3082	8,192	3-1-04	O	Mark.919	2,048	6-30-03	O
Matrix.750	4,096	11-22-03	O	Matrix.844	4,096	4-18-02	O
Matrix.LS.1820	8,192	8-26-01	O	Matrix.LS.1885	8,192	11-22-03	O
Matrix.Zelda.a	4,096	3-10-04	O	Matrix.Zelda.b	8,192	11-22-03	O
Matrix.Zelda.c	8,192	3-16-01	O	Matyas.644	45,700	11-22-03	X
Maya.4106	4,188	12-7-99	X	Maya.4108	8,192	11-22-03	X
Maya.4113	12,800	3-1-04	X	Maya.4114	8,192	11-22-03	O
Maya.4161	8,192	3-1-04	O	Maya.4206	8,192	10-4-02	O
Maya.4254	8,192	3-1-04	O	Maya.4608	8,192	10-4-04	X
Melder	46,080	6-27-03	O	Minit.b	10,752	4-27-04	X
MircNew	25,088	10-5-02	O	Mix.1852	4,096	5-30-00	O
Mockoder.1120	4,192	8-31-02	O	Mogul.6800	12,288	3-1-04	X
Mogul.6806	12,288	3-25-01	O	Mogul.6845	57,344	11-22-03	O
Mogul.7189	12,288	3-25-01	X	Mooder.a	8,192	4-3-02	O
Mooder.d	8,192	4-6-02	O	Mooder.f	14,452	8-19-03	X
Mooder.g	8,192	4-7-03	O	Mooder.i	8,192	5-1-03	O
Mooder.j	8,192	5-1-03	O	Morgoth.2560	2,560	11-22-03	O
Mystery.2560	130,544	12-8-01	O	NDie.2168	1,82,504	11-22-03	O
NDie.2343	1,82,725	11-22-03	O	Neoval	143,35	5-18-03	O
NGVCK.gen	3,584	10-5-04	O	Nicolam	57,344	4-27-03	O
Niko.5178	65,611	11-22-03	X	Noise.410	57,344	3-10-02	O
Opdoc.1204	1,23,448	6-28-03	O	Opdoc.1248	9,440	6-24-03	X
Oporto.3076	37,950	10-4-04	O	Padic	8,192	3-1-04	O
Paradise.2116	8,192	9-29-02	O	Paradise.2168	8,192	9-22-02	O
Parvo	80,093	3-1-04	O	Peana	8,192	3-1-04	O
Perrun.a	11,780	3-10-04	O	Perrun.b	5,636	7-11-02	O
PGPME	86,016	3-1-04	X	Pilsen.4096	4,096	3-1-04	O
Positon.4668	8,192	7-15-02	X	Qozah.1386	4,096	1-21-99	O
Qozah.3361	8,192	12-4-01	O	Qozah.3365	8,192	3-1-04	O
Qozah.3370	8,192	8-2-99	X	RainSong.3874	8,192	11-22-03	O
RainSong.3891	61,509	3-1-04	O	RainSong.3910	8,192	12-5-01	X
RainSong.3956	12,288	10-4-04	X	RainSong.4198	8,192	3-12-02	O
RainSong.4266	12,288	10-4-04	X	Ramdile	18,801	3-1-04	X
Razenya	8,192	10-25-03	O	Redart.2796	3,80,972	8-172000	X
Redemption.a	16,384	3-10-04	O	Redemption.b	16,384	3-1-04	O
Redemption.c	7,171	6-15-98	O	Refer.2939	36,352	3-1-04	O
RemEx	224,256	3-1-04	X	Revaz	8,192	3-1-04	O
Rever	32,768	3-1-04	X	Rhapsody.2602	221	10-4-04	O
Rhapsody.2619	8,192	3-1-04	O	Riccy.a	32,768	3-10-04	O
Riccy.b	1,72,032	10-28-01	O	Riccy.c	24,576	11-22-03	O
Rigel.6468	106,496	11-22-03	X	Rikenar.1480	8,192	9-9-98	O
Rivanon	3,584	6-26-03	O	Rufoll.1432	2,560	2-11-02	O
Rutern.5244	9,340	11-5-03	O	Ryex	8,192	3-1-04	O
Sadon.900	8,192	3-1-04	O	Sandman.4096	4,096	7-4-02	O

Table 9 continued

Virus name	Size	Date	Detection	Virus name	Size	Date	Detection
Sankei.1062	8,192	8-28-03	O	Sankei.1409	8,192	6-8-03	O
Sankei.1455	8,192	8-28-03	O	Sankei.1493	8,192	6-8-03	O
Sankei.1766	8,192	6-8-03	O	Sankei.1983	8,192	6-8-03	O
Sankei.3001	8,192	6-8-03	O	Sankei.3077	8,192	6-8-03	O
Sankei.3480	8,192	6-8-03	O	Sankei.3514	8,192	8-6-03	O
Sankei.3580	8,192	6-8-03	X	Sankei.3586	8,192	6-8-03	O
Sankei.3621	8,192	8-6-03	O	Sankei.4085	8,192	2-8-04	O
Santana.1104	81,920	12-4-01	O	Savior.1680	8,192	1-8-01	O
Savior.1696	12,288	5-18-01	O	Savior.1740	12,288	3-28-02	O
Savior.1828	20,480	8-6-01	O	Savior.1832	12,288	3-1-04	O
Savior.1904	12,288	12-4-01	O	Saynob.2406	5,120	5-15-03	O
Segax.1136	8,192	3-1-04	O	Segax.1137	8,192	3-3-03	O
Segax.1160	8,192	7-12-01	O	Sentinel.a	16,384	3-10-04	X
Senummy.1838	8,192	10-2-03	O	Seppuku.1606	8,192	9-2-02	O
Seppuku.2763	30,208	5-8-02	O	Seppuku.2764	30,208	3-1-04	O
Seppuku.4827	1,02,400	9-22-02	O	Seppuku.6834	12,288	7-12-02	O
Seppuku.6972	12,288	9-1-01	O	Seppuku.6973	12,288	7-12-02	O
Seppuku.9728	12,288	12-4-01	O	Shan.1842	4,096	4-27-02	O
Shown.538	4,096	4-6-02	O	Shown.539.a	4,096	3-10-04	O
Shown.540.b	4,096	5-19-03	O	Silcer	17,920	3-1-04	X
Slaman.a	24,576	6-28-03	O	Slaman.i	24,576	7-13-04	O
Small.1144	2,560	1-4-01	O	Small.1368	53,248	5-5-02	O
Small.1388	8,192	9-2-02	O	Small.139	94,208	7-17-02	O
Small.1393	8,192	1-2-02	O	Small.1416	16,699	8-7-02	O
Small.1424	8,192	1-2-02	O	Small.1468	8,192	4-6-03	O
Small.1700	4,286	8-5-02	O	Small.2218	96,526	9-2-02	X
Small.2280	96,588	12-31-00	X	Small.2560	8,192	4-25-03	O
Smog.b	12,288	10-2-03	O	Spelac.1008	4,096	11-17-02	O
Spreder	5,95,924	5-9-03	X	Staro.1538	8,192	3-1-04	O
Stepar.b	39,936	5-2-03	X	Stepar.dr	19,456	1-1-04	X
Stepar.e	65,536	5-5-99	X	Stepar.f	1,50,528	8-23-01	X
Stepar.g	1,37,216	8-23-01	O	Stepar.j	1,39,264	8-23-01	O
Sugin	1,47,456	9-19-02	O	Suns.3912	20,468	9-2-02	O
SWOG.based	4,096	6-13-02	O	Taek.1275	6,144	7-27-02	O
Tapan.3882	12,288	12-31-01	O	Team.a	4,096	3-10-04	O
Team.b	4,096	8-30-02	O	Team.c	4,096	9-22-02	O
Team.d	4,096	9-14-02	O	TeddyBear	2,560	3-1-04	O
Tenta.2045	10,240	5-9-02	O	Test.1334	67,072	9-2-02	O
This31.16896	51,202	5-8-01	O	Thorin.11932	16,384	3-1-04	O
Thorin.b	16,384	3-1-04	O	Thorin.c	16,384	10-23-99	O
Thorin.d	16,384	7-14-99	O	Thorin.e	16,384	10-23-99	O
Tolone	12,288	2-9-03	X	Ultratt	332	9-19-01	X
Ultratt.8152	12,288	3-1-04	X	Ultratt.8167	12,288	10-4-02	O
Undertaker.4887	12,288	11-22-03	O	Undertaker.5036.a	12,288	11-22-03	O
Usem.a	16,384	7-11-02	O	Usem.b	16,384	7-11-02	X
Vampiro.7018	18,432	3-1-04	X	Vampiro.a	16,896	3-10-04	O
VbFrm	28,672	7-26-02	O	VCell.3041	8,192	3-31-01	O
VCell.3468	8,192	8-29-01	O	VCell.3504	8,192	3-1-04	O
VChain	1,10,592	3-1-04	O	Velost.1186	8,192	9-19-02	O
Velost.1233	84,394	4-9-04	O	Velost.1241	56,963	4-22-04	X
Vorcan	8,192	10-4-04	O	Vulcano	12,288	3-1-04	O
Wabrex.a	8,192	3-10-04	O	Weird.10240	9,216	3-1-04	O
Weird.c	83,968	10-7-00	O	Weird.d	20,480	11-22-03	O
Wide.8225	16,896	7-30-02	X	Wide.b	12,288	8-31-02	O
Wide.c	12,288	8-8-02	X	Wolf.b	4,096	2-27-02	O
Wolf.c	8,192	10-4-04	O	Xorala	3,06,176	3-7-04	O
Xoro.4092	6,140	5-16-02	O	Yasw.1000	4,096	11-22-03	O
Yasw.924	4,096	12-29-00	O	Yerg.9412	28,672	5-25-02	O
Yerg.9571	16,384	11-22-03	O	Younga.4434	83,527	5-20-01	X
Zaka.a	2,809	11-1-04	X	Zawex.3196	32,768	9-22-02	X
ZHymn.a	88,064	4-5-01	O	ZHymn.b	90,112	8-23-01	O

Table 9 continued

Virus name	Size	Date	Detection	Virus name	Size	Date	Detection
ZHymn.Host	10,752	3-1-04	O	ZMist	86,016	3-1-04	O
ZMist.d.dr	28,672	3-1-04	X	ZMist.dr	28,672	10-4-04	X
Zombie	19,131	3-1-04	O	Zomby.17920	17,920	11-22-03	O
ZPerm.a	99,840	3-10-04	O	ZPerm.a2	73,728	11-13-00	O
ZPerm.b	70,144	3-1-04	O	ZPerm.b2	1,39,264	11-22-03	X

Win32 Virus Total Number: 499, Error number: 103, False negative: 0.2064 (approx. 21%)

Date indicates the virus's detected and caught date in form MM-DD-YY

Virus names are also the names of the test files. Size unit is Byte

Table 10 Normal executable program's virus check result by *VirusDetector*

Filename	Detection	Filename	Detection
dxwebsetup.exe		divx311.exe	
awsepersonal.exe		DVD2DIVXVCD_trial.exe	
paulp_en1.exe		adrenalin2.0.1.exe	
GOMPLAYER14.exe		sdvd190.exe	
csdl13.exe		HwpViewer.exe	
SSHSecureShellClient-3.2.9.exe		DivX505Bundle.exe	
klcodec220b.exe		SwansMP24a-WCP.exe	
DivXPro511GAINBundle.exe		NATEON.exe	X
WinPcap_3_1_beta_3.exe		ducp708_type3_free.exe	
wmpcdcs8.exe		Acrobat.exe	
iTunes.exe		pccmain.exe	
sicstusc.exe		Tra.exe	
acrodist.exe		java.exe	X
PCcpfw.exe		sicstus.exe	
Trialmsg.exe	X	Ad-Aware.exe	
javaw.exe	X	PCctool.exe	
splfr.exe	X	tsc.exe	
AdobeUpdateManage.exe		jpicp132.exe	X
Photoshp.exe		spmkds.exe	
conf.exe		policytool.exe	X
spmkrs.exe	X	unregaaw.exe	
CSDL.exe		kinit.exe	X
Powerpnt.exe		ssh2.exe	
Unwise.exe		csdlvw.exe	
klist.exe	X	Quicktimeplayer.exe	
Sshclient.exe		Wavtoasf.exe	
dialog_patch.exe	X	ktab.exe	X
ssh-keygen2.exe		Winword.exe	X
Directcd.exe		moviemk.exe	X
rmiregistry.exe	X	Tmntsrv.exe	
Winzip32.exe		Dreamweaver.exe	
msmsgs.exe	X	Scandisc.exe	
Tmoagent.exe		Excel.exe	
MSohtmed.exe	X	tmproxy.exe	X
iedw.exe		orbd.exe	
servertool.exe	X	tmupdito.exe	
iexplore.exe	X	PCClient.exe	
sftp2.exe		tnameserv.exe	X
uninstall.exe	X	keytool.exe	X
rmid.exe	X	scp2.exe	

Normal executable file's total number: 80, Error number: 24, False positive: 0.3 (approx. 30%)

If the result of detection is marked X, *VirusDetector* says this file is a virus-infected file, which means incorrect detection

References

1. Chantico: Combating computer crime: prevention, detection, investigation. McGraw-Hill, Inc, New York
2. Sophos: Top ten viruses and hoaxes reported to sophos in september 2005 (2005)
3. Yoo, I.: Visualizing windows executable viruses using self-organizing maps. In: Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS 2004), Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC-04) (2004)
4. Kohonen, T.: Self-organizing maps. Springer, Berlin Heidelberg New York (1995)
5. Haykin, S.: Neural networks: a comprehensive foundation, International Edition/2nd edn. Prentice Hall Englewood cliffs (1999)
6. Kohonen, T.: Self-organized formation of topologically correct feature maps. Biol. Cybern. **43**: 59–69 (1982)
7. Kohonen, T.: Self-organization and associative memory, 3rd edn. Springer, Berlin Heidelberg New York (1988)
8. Hinton, G., Sejnowski, T.J.: Unsupervised learning: foundations of neural computation. The MIT Press, Cambridge (1999)
9. Yoo, I., Ultes-Nitsche, U.: How to predict email viruses under uncertainty. In: Proceedings of the 23rd IEEE International Performance, Computing and Communications Conference, IPCCC 2004, Workshop of Information Assurance (WIA 04) (2004)
10. CERT: Cert/cc incident note in-99-03 cih/chernobyl virus. (1999)
11. Pfleeger, C.P.: Security in computing, International Edition, 2nd edn. Prentice-Hall International, Inc., Englewood cliffs (1997)
12. Kaspersky, E.: Virus analysis texts – macro viruses. (2000)
13. Esa Alhoniemi, Johan Himberg, J.P., Vesanto, J.: Som toolbox 2.0, a software library for matlab. SOM Toolbox team, Laboratory of Computer and Information Science, Finland (2002)
14. MATHWORKS: The mathworks, inc. MATLAB (2003)
15. KASPERSKY: Windows viruses (1994–2005)