Chapter 3

Immune Principle and Neural Networks-Based Malware Detection

Detection of unknown malware is one of most important tasks in Computer Immune System (CIS) studies. By using nonself detection, diversity of anti-body (Ab) and artificial neural networks (ANN), this chapter proposes an NN-based malware detection algorithm. A number of experiments illustrate that this algorithm has high detection rate with a very low false positive rate.

3.1 INTRODUCTION

Detection methods of viruses are generally divided into two categories. One is based on signature, which can be used to quickly detect a known virus but doesn't detect unknown and mutated viruses. As the number and classes of the viruses increase, the detection speed of this method will be slowed gradually down because the analysis of virus signature becomes very difficult and time consuming. To cope with this problem, IBM Anti-virus Group proposed an automatic extraction method of virus signatures even though it cannot extract the signature of an encrypted virus [1]. The other method is based on expert knowledge, with which one can construct a classifier to detect unknown viruses [2]. This method can discover some unknown viruses, but its false detection rate is high. To detect unknown viruses, IBM Anti-virus Group also proposes one ANN-based method to detect Boot Sector virus [3] only. Schultz and associates proposed a Bayes method to detect malicious executables, obtaining a good result [4].

Aiming at automation detection malicious executables, this chapter proposes a novel malware detection algorithm (MDA) [5] based on the immune principle and artificial neural networks (ANN) [6]. Extensive experiments show that the algorithm has a better detection performance than Schultz's method.

Artificial Immune System: Applications in Computer Security, First Edition. Ying Tan.

3.2 IMMUNE SYSTEM FOR MALICIOUS EXECUTABLE DETECTION

3.2.1 Nonself Detection Principles

To natural immune system, all cells of body are divided into two types of self and nonself. The immune process is to detect nonself from the cell set. To realize the nonself detection, the mature process of the T cell experiences two stages of selection, which are Positive Selection and Negative Selection. Inspired by these two stages, some computer scientists had proposed a few algorithms to detect anomaly information. For example, Forrest and associates proposed a Negative Selection Algorithm (NSA) based on the negative selection process in BIS for computer virus detection [7,8].

3.2.2 Anomaly Detection Based on Thickness

During the anomaly recognition process, cells of the immune system detect antigens and are activated. The activation threshold of immune cells is determined by the thickness of immune cells matching antigens. This property is very useful for improving the anomaly detection power and protecting immune system balance [9,10].

3.2.3 Relationship Between Diversity of Detector Representation and Anomaly Detection Hole

The main function of the anomaly detection utmost decreases the anomaly detection hole, which was efficiently resolved in the natural immune system (BIS). The diversity of the representation of MHC cells determines the diversity of antibody touched in surface of T cells. This property is very useful to increase the power of detecting mutated antigens and decrease the anomaly detection hole [11]. Inspired by this principle in Computer Immune System (CIS) [12] research, we can make use of the diversity of detector representations to decrease the anomaly detection hole.

3.3 EXPERIMENTAL DATASET

The experimental dataset is composed of 4481 executables, which includes 915 benign executables and 3566 virus programs. All files are scanned then classified by a virus cleaner tool. We collected all $*$.exe files during the windows 2000 operation system and some other application programs were installed in a computer. Through scanning these files by a virus killer, we constructed the benign executable set with 915 executables. The 3566 virus files are collected from the Internet and are also scanned as virus files by a virus killer, that is, the malicious executables set mainly consists of DOS virus, win32 virus, Trojan, Worm, and so on.

3.4 MALWARE DETECTION ALGORITHM

3.4.1 Definition of Data Structures

Definitions include:

B: binary code alphabet, $B = \{0, 1\}$.

$Seq(s, k, l)$: short sequence cutting operation. Suppose s be binary sequence, and $s = b(0)b(1)\ldots b(n-1)$, $b(i) \in B$, then $Seq(s, k, l) = b(k)b(k+1)\ldots b(k+l-1)$, k is the starting position of the short sequence in s.

$E(k)$: executables set, $k \in m, b$, m denotes malicious executable, b denotes benign executable.

E: all executables set, $E = E(m) \cup E(b)$.

$e(f_j, n)$: executable, $e(f_j, n)$ can be expressed as binary sequence that its length is n, and f_j is executable identifier.

l_d: detector code length.

l_{step}: detector generation step length.

d_l: detector, $d_l = Seq(s, k, l)$.

D_l: detector set, detector code length is l, $D_l = d_l(0), d_l(1), \ldots, d_l(n_d - 1)$, $|D_l| = n_d$.

3.4.2 Detection Principle and Algorithm

One executable can be regarded as a binary code sequence. Different binary code sequences of an executable decides that program has a different function. So, we might guess that using the binary sequence allows it to distinguish different programs, and then detect the function of that program. Based on the above principle, we constructed the detectors with short binary sequence and hoped this kind of detector can distinguish malicious executables from benign executables. We tried to build a set of short sequences that included benign executables' properties. These short sequences were used to construct the detector set D_l. Now, the question is how to generate these binary short sequences, that is, detectors, and how to detect nonself by using these binary short sequences.

This algorithm is based on the principle of nonself detection. First of all, the algorithm constructs a gene (G) that is used to generate detectors. The gene is composed of a gene unit, $e(f_j, n)$. These units are composed of benign executables. Here, G is constructed with $E_g(b)$, a subset of $E(b)$. Second, the algorithm extracts anomaly information according to the anomaly detection principle based on thickness. Third, according to the diversity of detector representation, the algorithm detects malicious executables with superior classifier. In summary, this algorithm includes three parts: detector generation, anomaly information extraction, and classification, whose block diagram is shown in Fig. 3.1.

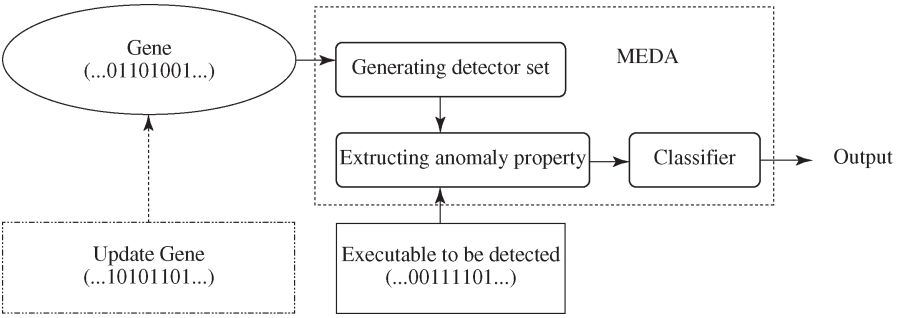

Figure 3.1 Block diagram of the malware detection algorithm.

3.4.3 Generation of Detector Set

3.4.3.1 Detector Code

Detector is encoded as a binary sequence, $d_l = Seq(s, k, l)$. The diversity of detector is realized by changing the code length l [13]. If code length $l = l_0, l_1, l_2, \ldots, l_m$, detector has m kinds, then detector set(D_l) also has m kinds.

3.4.3.2 Generation of Detector Set

Through partitioning the binary sequence of a program into equal lengths, detectors are generated. There are two parameters to be set before the detector set is generated. One is the generating step length l_{step}; the other is detector code length l_d.

3.4.4 Extraction of Anomaly Characteristics

3.4.4.1 Nonself Detection

Nonself detection is the first step of anomaly characteristics extraction. The nonself detection step finds short sequences that are not included in the self-set. Because we construct the detector set with a self short sequence, we use the Positive Selection Algorithm (PSA) [14] to detect nonself. Contrasting with NSA, PSA uses self-units as detectors. On the process of nonself detection, the short sequence not matched by any detector is nonself, otherwise, it is self. Its work process is shown in Fig. 3.2.

Algorithm 6 Detector Generating Algorithm

```
initialize l_step, l_d, k = 0.
repeat
   cutting e(f_k, n) from E_g(b).
   i = 0
   while i ≤ n − l_d − 1 do
      d = Seq(e(f_k, n), i, l_d).
```

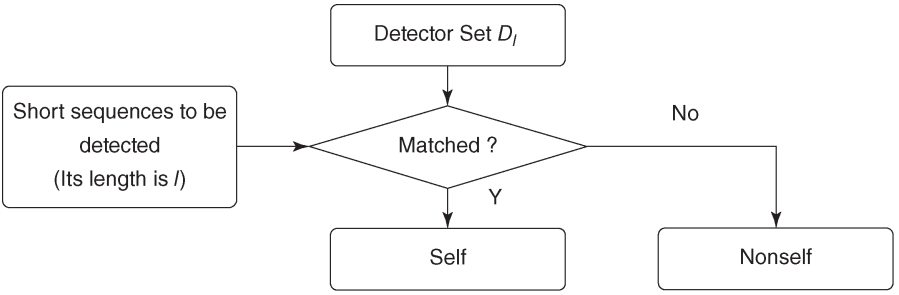

Figure 3.2 Diagram of the anomaly detection with PSA.

```
        if d ∉ D_{l_d} then
            D_{l_d} ← d
        end if
        i = i + l_step.
    end while
    k = k + 1.
until cutting e(f_k, n) is over. return D_{l_d}
```

3.4.4.2 Extraction of Anomaly Property Vector

According to the anomaly detection principle of the immune system based on thickness, we define the percentage of nonself unit number to file binary sequence as Anomaly Property. Anomaly Property is named as Nonself Thickness (NTh) in this chapter, and expressed as p_l. If detectors have m kinds, the file has an Anomaly Property Vector $P = (p_{l_1}, p_{l_2}, \ldots, p_{l_m})$.

Algorithm 7 NTh Extraction Algorithm

```
Open e(f_k, n).
Select l_step, l_d.
Set n_s = 0, n_n = 0.
i = 0.
while i ≤ n − l_d − 1 do
    s = Seq(e(f_k, n), i, l_d).
    if s ∉ D_{l_d}
        n_n = n_n + 1.
    else
        n_s = n_s + 1.
    end if
    i = i + l_step.
end while
p_{l_d} = n_n/(n_s + n_n). return p_{l_d}
```

To different detector set, this algorithm computes a separate NTh of the file. Finally, all NThs are combined to construct the Anomaly Property Vector $P = \left(p_{l_1}, p_{l_2}, \ldots, p_{l_m}\right)$.

3.4.5 Classifier

We use the Anomaly Property Vector as input of a BP neural network classifier, which consists of two layers. Anomaly Property Vector of the file is the input vector of this network. The number of neurons in the first layer is equal to the dimension of Anomaly Property Vector. The transfer function of input layer uses a Sigmoid-type function, while a liner function is chosen as the transfer function of the output layer of the BP neural network classifier [15].

3.5 EXPERIMENT

The first goal is to verify the detection ability of the malware detection algorithm for malicious executables. The experimental results are all scaled with false positive rate (FPR) and detection rate (DR). The second goal is to calculate the probability of the reducing detection hole with the diversity of detectors. Last, we compare our experimental results with Schultz's results [4].

3.5.1 Experimental Procedure

3.5.1.1 Partitioning Dataset

The whole dataset is divided into benign and malicious executables subsets, $E(b)$ and $E(m)$, respectively. $E(b)$ is composed of 915 program files and is divided into the generating detector dataset $E_g(b)$, which is the gene used to generate the detector data set, and the testing dataset $E_{test}(b)$. Here, $E_g(b)$ includes 613 program files and is constructed by randomly selecting from $E(b)$. The other 302 program files in $E(b)$ are elements of $E_{test}(b)$. In addition, $E(m)$ is composed of 3566 virus files.

3.5.1.2 Generating Detector Set

We select $E_g(b)$ as the gene of generating detectors, $ld \in 16, 24, 32, 64, 96$, and $l_{step} = 8$ bits. By using the detector generating algorithm, we can obtain five detector sets with D_{16}, D_{24}, D_{32}, D_{64}, and D_{96}, separately. In order to reduce the space complexity of detector savings, this algorithm uses Bitmap Index technology combined with Tree technology to store detectors.

3.5.2 Experimental Results

3.5.2.1 When to Use a Single Detector Set

Initially, we separately detect malicious executables with a single detector set of five kinds. The experimental result is shown in Figs. 3.3 and 3.4.

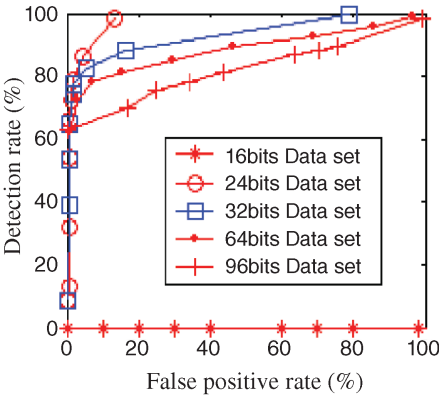

Figure 3.3 ROC curves for five kinds of detector sets.

When $FPS = 0$ and $l_d = 96$, it can be seen from Figs. 3.3 and 3.4 that the best detection effect can be obtained, and the DR becomes worse as the length of l_d becomes longer. The DR becomes better as the FPS becomes worse. When $FPS \geq 1\%$, the detection rate is the highest when $l_d = 24$. Meanwhile, with the length of l_d, the DR will be worse. On the other hand, when $l_d = 16$, the DR is 0 because the number of detectors reaches 2^{16}. When $l_d = 96$, the DR is almost the same regardless of FPS values.

3.5.2.2 When to Use Multi-Datasets and BP Network Classifier

As described in Section 3.4, this experiment uses multi-datasets to detect benign and malicious executables. Here we don't use the D_{16} dataset because the DR is 0 for this

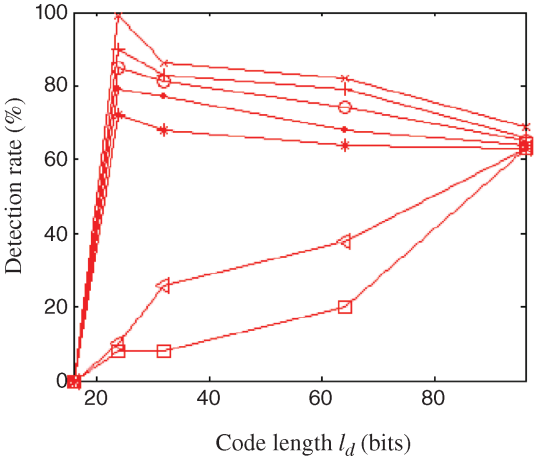

Figure 3.4 When FPR is constant, the curves of DR with l_d. From bottom up, the FPR is 0%, 0.5%, 1%, 2%, 4%, 8%, and 16%, respectively.

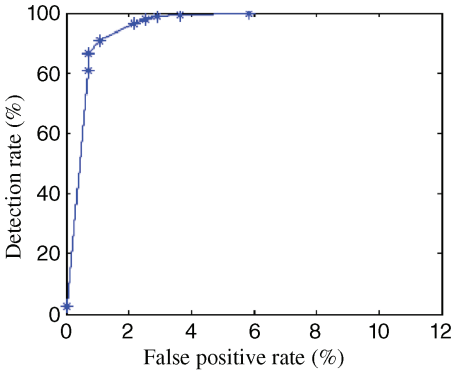

Figure 3.5 Experimental ROC curves with BP network classifier.

dataset. At the same time, we set the upper limit of l_d to be 96 because the DR is almost the same when $l_d = 96$. So, in this experiment, we selected the four datasets of D_{24}, D_{32}, D_{64}, and D_{96}, as anomaly detection sets, and use them for extracting the Anomaly Property Vector. Finally, a BP network with two layers is used as a classifier. In the classification process, we randomly select 30 percent files of $E_{test}(b)$ to train the BP network classifier, and use the other to test the performance of the anomaly detection algorithm. The experimental results are shown with ROCs in Figs. 3.3–3.5 [16].

3.5.3 Comparison With Matthew G. Schultz's Method

3.5.3.1 Comparison of Datasets

Table 3.1 compares experimental sets.

3.5.3.2 Comparison of Algorithmic Performance

Comparison of algorithmic performance includes two aspects—detection rate and the complexity of algorithms.

1. **Detection rates of algorithms:** Known from Schultz's experimental result [4], using the Naive Bayes approach, the DR can reach 97.43 percent when FPR is 3.80 percent. As for MEDA, when FPR is 2 percent, the DR can reach the same value. When to use Multi-Naive Bayes, the detection rate of the algorithm obtains the highest value of 97.76 percent. But for MEDA, the highest DR can reach 99.50 percent. The detailed comparisons are shown in Fig. 3.6.
2. **Complexity of algorithms:** The complexity of algorithms includes both computational complexity and space complexity. The space complexity is mainly about the space size to store detectors and probability information ($P(F_i/C)$) of two kinds of algorithms. The malware detection algorithm used $4000MB$ to store detector set (shown in Table 3.2), while Schultz used about $1\ GB$ to store his probability information [4].

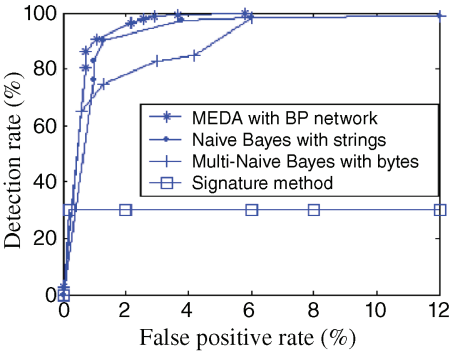

Figure 3.6 Comparison of experiment results for several algorithms.

Table 3.1 Comparisons of Experimental Data Sets

Data set	Operation system	Collection method	Size of data set	$E(b)$	Number	$E(m)$ kinds
M.G. Schultz's data set	Windows	Collecting all $*$.exe file to construct E(b), and collecting E(m) in Internet	4266	1001	3265	Trojan, virus
Our data set	Windows	As above	4481	915	3566	Trojan, worm and virus

As for the computational complexity, the main operational units in MDA include generating detector, matching short sequence, computing nonself thickness, and classification by BP network. The main operational units of Schultz with Bayes include generating probability information, searching conditional probability $P(F_i/C)$ [4], computing $P(C)\prod_{i=1}^{n}P(F_i/C)$, and classification by probability. In this chapter for simplicity, we won't think about the classification operations. Suppose file size of training dataset is l_{train} bytes, and the size of the file to be detected is l_{test} bytes, the computational complexities of the two algorithms are shown in Table 3.3.

Table 3.2 Generation of Detectors

Detector code length l_d	16	24	32	64	96
$\lvert D_{l_d}\rvert$	65536	10,931,627	8,938,352	12,768,361	21,294,857
Methods to store detectors	Bitmap index	Bitmap index	Tree	Tree	Tree

Table 3.3 Comparison of Computational Complexities of Two Algorithms

Algorithms	Operation type 1		Operation type 2		Operation type 3	
	Name	Operation number	Name	Operation number	Name	Operation number
MDA	Generating detectors	l_{train}	String matching	$\leq 80*l_{test}$ [1)]	Computing non-self thickness	$4*l_f$ times addition
Bayes	Generating probability information	$\gg l_{train}$	Searching $P(F_i/C)$	Rest with the number of $P(F_i/C)$	Computing $P(C)\prod_{i=1}^{n}P(F_i/C)$	l_f times float multiplication

3.6 SUMMARY

It turns out from a number of experiments that the detector generated with executable binary short sequences can be used to detect malicious and benign executables. When using a single detector set to detect malicious executables, the best performance with the DR reaches 80.6 percent when FPS is 3 percent. Meanwhile, if a longer code length of detector is used, the DR of single detector set becomes lower. The best detection performance of our algorithm is obtained when a multi-detector set is used to detect malicious executables with 97.46 percent DR when FPS is 2 percent. This verified our claimed principle that diversity of detector representations can decrease the anomaly detection hole, and further it also validated nonself detection and detection based on thickness.

REFERENCES

1. Lo, R.W., Levitt, K.N., and Olsson, R.A. (1995) Mcf: A malicious code filter. *Computers & Security*, **14** (6), 541–566.
2. Arnold, W. and Tesauro, G. (2000) Automatically generated win32 heuristic virus detection, in Proceedings of the 2000 International Virus Bulletin Conference, September 2000, pp.51–58.
3. Wikipedia, Boot sector viruses.
4. Schultz, M.G., Eskin, E., Zadok, E., and Stolfo, S.J. (2001) Data mining methods for detection of new malicious executables, in Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on, IEEE, pp. 38–49.
5. Guo, Z., Liu, Z., and Tan, Y. (2004) An NN-based malicious executables detection algorithm based on immune principles, in *Advances in Neural Networks-ISNN 2004*, Springer, pp. 675–680.
6. Dayhoff, J.E. and DeLeo, J.M. (2001) Artificial neural networks. *Cancer*, **91** (S8), 1615–1635.
7. Forrest, S., Perelson, A., Allen, L., and Cherukuri, R. (1994) Self-nonself discrimination in a computer, in Research in Security and Privacy, 1994. Proceedings, 1994 IEEE Computer Society Symposium on, IEEE, pp. 202–212.
8. Guo, Z., Tan, Y., and Liu, Z. (2005) Non-self detector generating algorithm based on negative selection principle. *Journal of Chinese Computer Systems*, **26** (6), 959–964.

9. Guo, Z., Tan, Y., and Liu, Z. (2005) Malicious executables detection algorithm research based on immune system principles. *Journal of Chinese Computer Systems*, **26** (7), 1191–1195.
10. Guo, Z., Liu, Z., and Tan, Y. (2005) Detector generating algorithm based on hyper-sphere. *Journal of Chinese Computer Systems*, **26** (12), 1641–1645.
11. Tan, Y. and Guo, Z. (2005) Algorithms of non-self detector by negative selection principle in artificial immune system, in *Advances in Natural Computation*, Springer, pp. 867–875.
12. Somayaji, A., Hofmeyr, S., and Forrest, S. (1998) Principles of a computer immune system, in Proceedings of the 1997 Workshop on New Security Paradigms, ACM, pp. 75–82.
13. Tan, Y. (2006) Multiple-point bit mutation method of detector generation for SNSD model, in *Advances in Neural Networks-ISNN 2006*, Springer, pp. 340–345.
14. Dasgupta, D. and Nino, F. (2000) A comparison of negative and positive selection algorithms in novel pattern detection, in Systems, Man, and Cybernetics, 2000 IEEE International Conference on, IEEE, vol. 1, pp. 125–130.
15. Ruan, G. and Tan, Y. (2007) Intelligent detection approaches for spam, in Natural Computation, 2007. ICNC 2007. Third International Conference on, IEEE, vol. 3, pp. 672–676.
16. Crawford, R., Kerchen, P., Levitt, K., Olsson, R., Archer, M., and Casillas, M. (1993) Automated assistance for detecting malicious code, *Tech. Rep.*, Lawrence Livermore National Lab., CA (United States).