International Journal of the Physical Sciences Vol. 5(15), pp. 2236-2247, 18 November, 2010
Available online at http://www.academicjournals.org/IJPS
ISSN 1992 - 1950 ©2010 Academic Journals

Review

A biological model to improve PE malware detection: Review

Saman Mirza Abdulalla[1]*, Laiha Mat Kiah[1] and Omar Zakaria[2]

[1] Department of Computer System and Technology, Faculty of Computer Science and IT, University of Malaya, 50603 Kuala Lumpur, Malaysia.
[2] Department of Computer Science, Faculty of Defence Science and Technology, National Defence University of Malaysia, 57000 Kuala Lumpur, Malaysia.

Accepted 29 October, 2010

Malwares control computer systems by infecting system files. They take advantage of system compatibilities to ensure their survival from one version to another. The structure of the windows portable executable (PE) files between available versions of the windows operating system (OS) makes these files an easy target for malwares. Fields and codes of clean PE files are modified and changed after infection. Checking both changes and modifications is necessary to detect malwares with a minimum false alarm rate. This paper reviews models that propose to detect PE malwares. It discusses PE structure and identifies the fields and locations that are vulnerable to malwares. It also explains the use of the human immune system and co-stimulation signals as a way to build a biological model for improving the ability of PE malware detection systems.

Key words: Malware detection, false alarm, PE files, immunity system, co-stimulation signals.

INTRODUCTION

Malware is a generic term used to define different types of attack and threat codes that penetrate and infect computer systems (Bradfield, 2010). Most malwares act as an application inside a computer system (Jajodia, 2009). Applications are targeted by anti-malware software to detect and prevent their activities. Detection and prevention are difficult because malwares change their behaviors and structures with every new generation (Szor, 1998).

Periodically, technical reports from detection system vendors warn about new malwares, displaying charts and graphs to show how the number of malwares has increased over networks. A report from Symantec (2010) shows that the number of malwares has reached millions (Security, 2010). In addition, numerous zero-day malware infections are recorded and reported every day (Security, 2010). A 2010 report from McAfee shows that up to 55,000 new malwares are reported every day (McAfee, 2010).

Current detection systems and anti-virus programs scan computer systems to look for malware instances that are already known (Clemens Kolbitsch, 2009). The non-dissection malware codes cannot be captured by available commercial detection systems. This means that the valid detection systems are able to recognize only the known patterns of malwares (Security, 2010). However, for unknown malwares, malware analysts should extract the behaviors and codes of malwares, after which they should update detection systems. Only then could such detection systems function properly (Zakaria, 2009).

Although some anti-malware applications that depend on behavior detection techniques have the ability to predict the presence of some malwares, they still face two main challenges. First, available anti-virus programs are only able predict the presence of new malwares whose behaviors are closer to behaviors of known malwares (Jajodia, 2009; Security, 2010). This means that if there is a great difference between known and unknown behaviors, such a technique would likely fail to detect unknown malwares. With this inadequate ability to detect new malwares, the second undesirable

*Corresponding author. E-mail: saman1969@siswa.um.edu.my.

characteristic of the behavior-based detection system is the prevalence of false alarms, which only annoys users (Bradfield, 2010).

This paper reviews the modifications and changes that malwares inflict on system files, as well as how they control the system's execution and perform payloads and functionalities. This paper also tackles malware call functions of operating systems for execution and how these calls are traced to detect the execution behaviors of malwares. The work needs to include some explanations about biological co-stimulation and to study windows portable executable (PE) file structure to support proposing a biological model that could improve Malware detection and eliminate the False Alarm.

Malwares and executable file Infectors

Malicious code activities and functionalities infect computer systems every day. Various ways have been developed by virus writers to penetrate computer systems; however, prevention proves difficult. Moreover, current viruses have different types of payloads to perform different functionalities that challenge detection (Bradfield, 2010). Many types of malicious codes perform different activities; however, this paper will focus on just three types:

(a) Virus: A computer program that is usually hidden within another seemingly innocuous program, produces copies of itself and inserts them into other programs, and usually performs a malicious action (Zakaria, 2009).
(b) Worm: A small, self-contained, and self-replicating computer program that invades computers on a network and usually performs a destructive action (Khaled et al., 2010).
(c) Trojan Horse: A seemingly useful computer program that contains concealed instructions which, when activated, performs an illicit or malicious action (Bradfield, 2010).

This paper has chosen these three types of malicious codes because they represent 85% of the total infection cases recorded in annual reports for 2009 to 2010 issued by antivirus software vendors (McAfee, 2010; Security, 2010). In addition, the available antivirus programs still do not have the ability to recognize all types of malwares and detect zero-day malwares (Filiol et al., 2006; Sami et al., 2010).

The execution of such types of malwares is similar to the execution of any normal applications or programs that run under Windows OS. Malwares use many Windows functions stored in Kernel mode and user mode called Application Programming Interface (API) (Oney, 2002). To call these functions, malwares should have the physical addresses of the needed APIs, which cannot be obtained easily, and which Windows OS will not simply provide (Willems et al., 2007). Thus, malwares find ways to collect these addresses from the Windows OS.

Malwares are programmed to know that each normal application that runs under Windows OS has a predefined list of API names and addresses (Cheng, 2009). The listed API is imported by the application during execution or exported to other Windows applications (Oney, 2002). Malwares attack these PE applications to collect API addresses and control the execution of infected applications (Willems et al., 2007). They change certain fields and locations to direct the execution of the normal application PE to their codes, and then return the execution control to normal after performing their functionalities (Clemens, 2009). They also modify the list of needed API functions to include other functions required during code execution (Schmidt et al., 2009).

The idea to insert codes in executable files originated in 1986, when a programmer named Ralph Burger found that codes could be inserted into DOS executable file. He programmed a virus, called "Vir Dem," with an executable infection capability. The virus has the ability to infect .com executable file under MS-DOS (Szor, 1998).

When the Windows OS and Windows NT were released, most MS-DOS executable viruses failed to replicate under the new system. Only a few viruses were able to do so, such as Yankee Doodle, a very successful old Bulgarian virus (Merkel et al., 2010). Virus writers faced the new challenge of investigating the new operating system and creating new DOS executable viruses and boot viruses, with special attention to Windows 95 compatibility (Paquette, 2000). Since most virus writers did not have sufficient in-depth knowledge of the internal mechanisms of Windows 95, they looked for shortcuts to enable them to write viruses for the new platform (Shevchenko, 2007). They quickly found the first one—macro viruses, which are generally not dependent on the operating system or hardware differences (Skoudis and Zeltser, 2004). The first system-dependent virus, Boza, which was compatible with Windows 95, appeared on the same year that Windows 95 was launched. This virus was written by an Australian group called VLAD (Piccard, 2005)

A new system-dependent virus called Cabanas virus was later written. Developed by Jaky/29A at the end of 1997, the virus can infect Windows 95. Later, it was found to have the capability to infect other new Windows platforms, such as Windows 9X, Windows NT, and Win32. Soon the compatibility dream of the Windows platforms became a nightmare (Szor, 2006). The key role of Windows compatibility returned to the common file format or PE file format. This also makes it possible for viruses to jump from one 32-bit Windows platform to another easily, as Cabanas did (Chappell, 2006).

In May 2000, the number of executable infectors reached more than 500. However, the most important Win32 development by virus writers was in Win32 worms (Higgins et al., 2003).Worms can replicate, propagate, and infect without a user interface (Szor, 2006). The virus

writers moved from virus developers toward networked worm development. Moreover, techniques such as “code encryption” have made signature detection procedures more difficult (Paquette, 2000). In addition, most Win32 infectors are unable to save the “entry point” of the PE files; therefore, curing the infected files became difficult, or even impossible (Ször, 2000). These activities prompted virus researchers to spend more efforts on PE infectors.

The encryption techniques that hide signatures by encrypted PE viruses became more developed. Virus writers use polymorphic techniques to change the signature of a PE virus each time it infects a new file (Zakaria, 2009). The virus Win32.Driller, which was first reported in 2003, uses the Polymorphic techniques to hide its signature. The virus infects PE files that have .exe, .scr, and .cpl filename extensions. When run, the virus infects these files in current Windows and in Windows system directories (Tyagi and Vyas, 2008). The polymorphic technique gives the viruses the capability of changing their signature with each new infection (Xu et al., 2004). In 2004, Symantec released a report showing that it received 17,500 PE infector samples (Turner et al., 2004). Moreover, many strong malwares, such as Sassor and Win32 virus 1408, were released, and the types of system files that could be infected grew in number. Many data files of several antivirus programs were targeted and deleted by PE malwares (Fosnock, 2005).

In 2005 to 2006, the number of PE infectors and malwares increased. Records show that the number of malwares in 2006 increased by 41% from 2005. Records further indicate that approximately 1,500 malwares are being recorded every month, including worms that are propagated through e-mail. In 2006, another malware, Virus.Win32.Saburex.a, was recorded, which has the capability to use a technique called Packer (Turner et al., 2006). Packer is a software or a technique that is typically used to minimize the size of files or data that are transferred over networks or through e-mails (Wei and Ansari, 2008). Unfortunately, virus developers misused this technique to hide malicious codes inside PE files. The problem with this technique is that malicious codes cannot be scanned until the file is executed and the image of the PE is saved to the memory. In addition, scanners cannot mark any packed files as malwares because packers are initiated as normal software for files and data compression purposes (Wei and Ansari, 2008).

Increasing the complexity of PE malwares was come in parallel with their steady increase in number. Approximately 45,690 new malwares were recorded in November 2008 and 46,014 in January 2009 (Coorp, 2008). Antivirus and detection system programs should be aware of all these techniques and have adequate information about the behaviors of malwares to properly detect them. Complex malwares have many behaviors that are similar to those of clean files. This leads to increase the false alarm rate. Figure 1 shows the status of false alarm rate of (2010) version for antivirus detection softwares (AV-Comparative, 2010).

Files should be thoroughly checked to distinguish a clean file from an infected one with zero error rates. This work expected such improvement by inserting a double checking process in the detection systems, as Immune System does.

Malware detection models

Most reviewers and researchers agree that the significant problems in current malware detection models have to do with the accuracy of prediction and the ability to detect unknown malwares (Bradfield, 2010). There are many considerations for improving the ability of antivirus systems. Current signature-based antivirus programs still cannot detect unknown malwares, whereas behavior-based detection systems are accompanied by the annoying false alarm (Jajodia, 2009; Security, 2010).

Therefore, the time has come to go deeper into the infected file, and to open and see, as the surgeons say, what an external scan cannot identify (Cheng, 2009). Most infected files are the executable and system files; therefore, researchers aim to study the structure of these PE files, determine the models that can compute the differences between a clean and an infected case, and refine malware detection (Khaled et al., 2010).

Using classical file scanning, the researchers find difficulty in dealing with the sophisticated techniques used by malwares, such as polymorphic and metamorphic techniques (Xu et al., 2004; Mori, 2004). Xu et al. (2004) introduced a model that traced and analyzed API sequence for applications, using the API sequences as signatures to detect new strains of known polymorphic viruses. Xu et al. (2004) relied on a hypothesis that polymorphic viruses can change their signature; however, the sequence of their API did not change. The model can detect instances of known polymorphic viruses and can search for similarities to detect unknown strains. However, there are some malwares using the metamorphic techniques that have the ability to reorder the sequence of API calls (Mori, 2004). During the evaluation, optimization is needed to improve new sequence generation. Including more than one feature, rather than considering just the API behavior calculation feature, is necessary. With more than one feature, the analysis result of the second feature will support or correct the first one.

The same API sequence topic was proposed by Miao Wang (2009) to build an abnormal intrusion detection system. They saw that the domain used by Xu et al. (2004) does not cover all types of APIs in case new malwares are applied to the detection system. They trained their proposed model to detect abnormal API

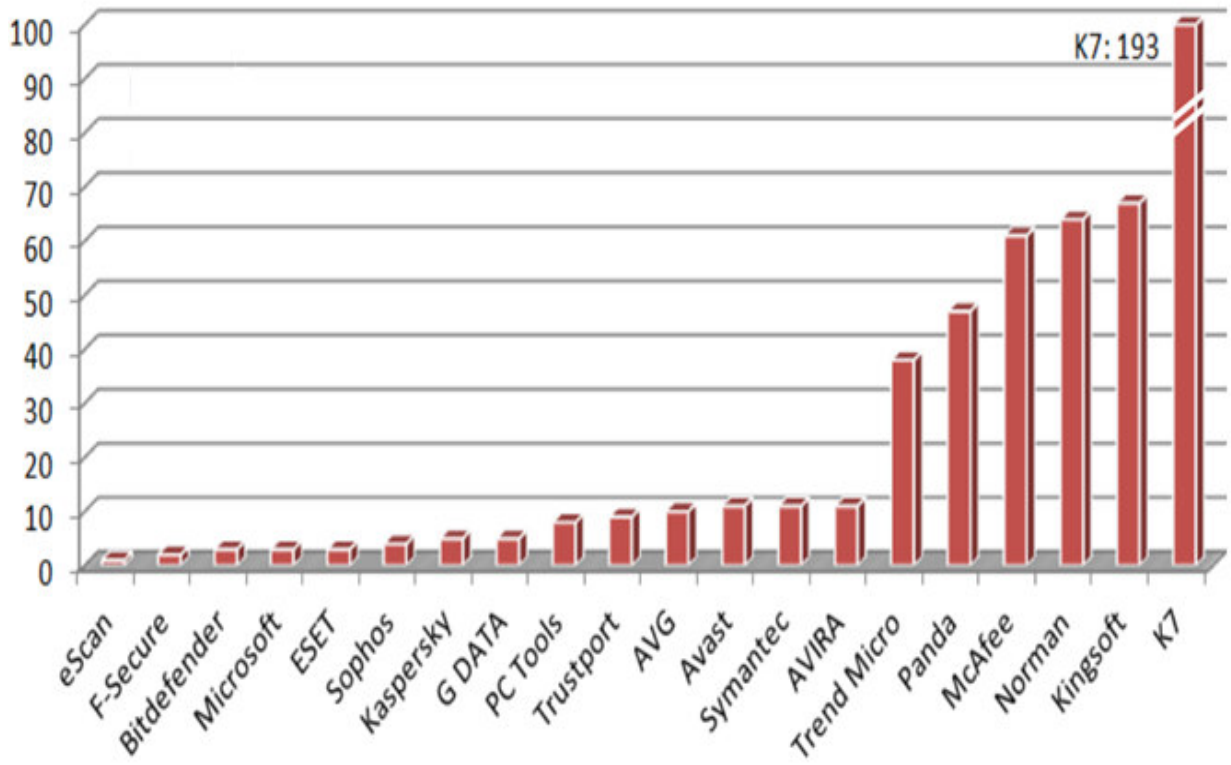

Figure 1. False alarm rate for available detection systems.

sequences using both native and user mode API functions, and found that monitoring the user API calls is not sufficient to trace new malwares. This is because many malwares can call API directly. Malwares can expand their domain to include all API functions available in the computer system in the training vectors, while the window size of trained vectors remain the same six elements used by Xu et al. (2004). This enables new APIs within each vector to be seen as malwares that somehow become included in the training set of normal vectors. Distinguishing such vectors from normal vectors is difficult and likely leads to a high false alarm rate because there are many shared APIs between normal vectors and abnormal vectors in the training set.

The API functions required by a PE file execution are saved in the import table field (Essam, 2008). Malwares simply make a few changes in the API calls inside the “import table” of the PE by inserting the new API functions needed to execute its payload (Szor, 2006). The percentage or degree of changes in the API list in import table shows if the file is infected or is still in the correct status. This concept was used by Karmer and Bradfield (2009) and (Bradfield, 2010) as a hypothesis to describe malwares. The hypothesis states that any changes in the correct software affect the correctness measurement (that is, degree of changes); therefore, malwares may be determined according to the degree of correctness. The process involves measuring the normality of a program that a malware requires to defeat scanning programs. Such models are very sensitive to any changes and have a high false alarm rate (Szor, 2006). They need verification to control and minimize this rate, and further investigate the suspected PE files.

The PE file anatomy is not a new topic; in fact, it has been used by many detection models to check for the presence of malware codes or analyze malware behavior (Father, 2004; Mori, 2004; Jajodia, 2009). (Khaled et al., 2010) built an AMI model based on clonal selection algorithm. They used the API call sequence to classify malwares, using compression between some algorithms, with acceptable results. They concluded that the false alarm still exists.

Many techniques are applied to the API call sequence to detect or classify malwares accurately. Sam et al. (2010) used data mining techniques to analyze the API call sequences. Although, they claimed that these techniques minimize the false alarm, minimization could be further improved and eliminated if they involved changes behavior of malwares as features for verification.

The verification method used by (Cheng, 2009) applied probability rules such as conditional probability. They classified the contents of PE fields into normal and abnormal behavior groups, and used the same API domain to identify the normal and abnormal probability values. They also used both values to calculate the

conditional probability on normality, and finally compare the result with a predefined threshold. However, no verification of normality or abnormality results is available before applying the conditional probability. In addition, the verification is built into the same domain. This is similar to analyzing two half stories from the viewpoint of one type of data. Building a verification process inside the malware detection system is necessary. The principle of verification is required to make a case meaningful or clear, which can be achieved in different ways.

Towards this end, determining what malwares do inside a PE file would be desirable, and discovering how modifications of the API sequence or API list in import table can confirm this information. Malwares make changes in several locations inside the PE files to obtain control of the program execution (Szor, 1998; Szor, 2006). The system should record the changes to these entry point locations made inside the PE files by malwares. These changes could be considered as a starting point to build a malware detection system (Mori, 2004). (Nachenberg, 2001) explained that most possible locations that a malware infects are changed. His model scanned these locations and sent the results to an emulation process to detect malwares. This action could support the verification of what an API sequence scanner will decide.

Biological models to detect malwares

In previous years, many biological models have been proposed to improve computer security systems. Most of them try to mimic the human immune system because of its strong capacity to defend against viruses and bacteria. Specifically, the immune system has the capability to discriminate between self- and non-self cells, and to update itself to launch unknown foreign cells inside the human body (Julie et al., 2010).

Forrest et al. (2002) proposed the self- and non-self discrimination process. In his research, Forrest showed how problems of computer security could be solved by making the computer system recognize the normal files and distinguish them from the abnormal ones. Other researchers have used mathematical approaches to prove his hypothesis. Theoretically, they obtained good results; however, no explanations or details have been provided regarding how this proposal could be validated. When an infected file that looks like a normal one is executed, for a few seconds or less than a second, it becomes harmful as it achieves its functionalities.

The normal file is considered as abnormal after insertion of malicious codes into normal one by a malware (Bradfield, 2010). The detection systems and antivirus programs should recognize the binary representation of these malicious codes or their execution behaviors. D'haeseleert et al. (2002) proposed the negative selection algorithm to build detectors that can recognize unwanted patterns. To build these detectors, they focused on information taken from the self set without referencing any information about the non-self set. This has led to an increase in the computational cost of this algorithm because the number of detectors is exponentially related to the size of the self set. Even if the relation improved linearly, as shown by Stibor et al. (2005), the number of required detectors remains too large; in addition, these are not guided detectors.

Most immune system-based models that work using pattern recognition utilize either clonal selection algorithm, negative selection algorithm, or the danger method (Srinivasan, 2009; Jieqiong et al., 2010) to build detectors. However, the discrimination process in the immune system is completed and confirmed only when the co-stimulation signal is found. A survey conducted by Bo-yun (2006) classified the applications that could be achieved by the artificial immune system. The survey included models that propose to build different security and detection systems. The algorithms used by detection models are built upon the processes that were already used to generate perfect detectors. These models presuppose the cases to be already abnormal and then simply attempt to create suitable detectors for the abnormality. The authors of all the proposed work agree that the key task of any AMI is discrimination; however, they built their models on algorithms that occurs during the proliferation stage, which comes after a confirmation signal from the co-stimulation process has confirmed the discrimination between self and non-self (Smith, 2006). Without the verification and confirmation process, the negative selection algorithm generates a false alarm. In their investigation, Kim and Bentley (2001) used NSA in the anomaly detection system. They found that this algorithm can work similarly to a filter for anomaly cases and cannot generate competent detectors. The result they obtained implicitly shows that models that mimic NSA are expected to have a high false alarm rate; thus, they proposed to extend the NSA model.

Other researchers, such as Stibor et al. (2005), proposed that the detection set in NSA should contain negative and positive elements in order to work properly. They suggested applying two classes to the support vector machine (SVM) to build an anomaly detection model. The suggestion did not mention any characteristics of the features of elements in each class, thereby affecting the accuracy of the model. The self and non-self discrimination by NSA was reviewed in detail by Aickelin et al. (2004). They explained many suggestions made by other researchers and showed the ability of AIS in the field of “intrusion detection system”. They encouraged doing further research in this area, applying a wider data set and samples in training and testing stages to minimize the false rate obtained during discrimination.

Apparently, the filtration of self from non-self performed by NSA needs optimization to improve the accuracy of the detection procedures. Researchers have worked to mutate

elements to predict new patterns. Towards this end, the clonal selection algorithm proposed by many researchers has been widely used as an extension and an improvement to the NSA. The algorithm has been described by Yu and Hou (2004) as a practical and simple tool for simulation and experiments. The steps of this algorithm, described biologically by Edge et al. (2006) as the self-organization, adaption, diversification and then positive selection and negative selection of an element, relate to self and non-self. The algorithm was investigated by De Castro and Timmis (2002), who showed how active detectors could be used to activate other detectors and maintain the ability of detection systems. The clonal selection algorithm reduces the time spent on detection. The algorithm achieves mutation to increase the affinity degree of matching, and allows the pattern recognition to be faster when the next infection occurs (Dasgupta, 2007). In addition, the mutation of the clonal selection algorithm helps the prediction of new patterns and new computer viruses within the range of the training domain. Biologically, this stage depends on the co-stimulation signal that has already identified a foreign cell as a non-self (Khaled et al., 2010).

Another survey done by Hart and Timmis (2008) on applications was inspired by the artificial immune system where anomaly detection and computer virus is one of the minor applications. They stated that problems in all research studies come from the information used to represent training sets, such as depending on one source in building set and how many elements should be inserted. The studies attempted to solve the issue of fitting the detectors with some anomaly or virus cases. This implies that unknown cases are already considered as viruses or anomaly cases. All these issues are defined from the viewpoint of negative selection and clonal selection. However, Aickelin and Cayzer (2002) noted that negative selection algorithm has a false alarm rate, and clonal selection algorithm has many shared areas with self and non-self sets. He tried to propose the Danger Theory to overcome the problems of previous algorithms. Matzinger's Danger Theory debates the self and non-self point of view (Kim et al., 2005). The theory agrees that the discrimination of self and non-self should occur, then negative selection and clonal selection algorithms can be achieved correctly. However, all programmers who used Danger Theory agree that this theory can work correctly only if the data set is limited (Glaser and Strauss, 2007).

As summarized in Table 1, the subject of using a biological model to promote security and analyze malwares is not new. However, there is a biological phenomenon in "human immune system" activities, called co-stimulation, which has not inspired work in security issues. This phenomenon is responsible for improving the accuracy of discrimination between self and non-self. This activity is responsible for not attacking self cells as non-self cells.

Immunity System Co-stimulation (ISC): Biological verification

The human body's immune system is an excellent defense system. It performs complex activities to keep the body clean from antigens, and adapts itself to detect and memorize unknown antigens (Health, 2003). Its key role is to distinguish between the normal (self) cells and abnormal (non-self) cells. The distinguishing process is achieved through cooperation between immune cells with the presence of some enzymes that work as communication signals between those cells (Hofmery, 2000). These signals control the activities of the immune system. They direct the defense process correctly and, in perfect situations, they instruct the immune cells in performing their functions, such as the proliferation process, when a specific antibody has been generated and memorized for an antigen (Naik, 2003).These signals are controlling the activities of the immune system. They are directing the defense process correctly and only in perfect situations they instruct immune cells to do their functions, such as the proliferation process, when a specific antibody has generated and memorized for an antigen (Naik, 2003).

As a first response, B-cells will engulf a suspected body and analyze it. Pieces of engulfed body as activate the major histocompatibility complex (MHC) in peptides on the surface of B-cell. The MHC rising in the B-cell signals to two types of T helper cells (that is, Th- CD+4 and Th- CD+8) to be stimulated the MHC (Michael, 1997; Naik, 2003).

When receptors of Th- CD+4 are activated with MHC, the first signal (Signal 1) detecting an abnormal case is satisfied. The degree of activation differs as not all receptors have the same shape as MHC. The degree of such activation represents the affinity. The Th-cell will bind with the MHC protein in another form using (CD+8) to confirm (Signal 1). The incorrect activation of (CD+8) Th-cell will not generate the confirmation signal (Signal 2). This means that Signal 1 is generated incorrectly and the engulfed B-cell will be marked as anergic cells (Health, 2003). However, correct activation will result in the co-stimulation signal. In this situation, the immune system will decide to build an arsenal of a certain type of antibody and killer cell through the proliferation of B and T cell to kill the antigens, thereby cleaning the body, and to memorize the built antibodies (Michael, 1997). Figure 2 illustrates the co-stimulation and its effects (Naik, 2003).
Co-stimulation signals, sometimes called two-signal messages, come from simultaneous activation of two different Th- cell types with an antigen (Naik, 2003). This

Table 1. Summary of malware detection system's works.

References / Researchers	Method used	Contribution	Analysis
(Forrest et al., 2002)	First one used mathematical approach to design biological model for detection system. They depend on negative selection algorithm.	They proved that malwares could be detected even they are unknown.	Their work needs to be validated.
(D'haeseleer, 2002)	They used negative selection algorithm to build a biological model.	They proved NSA statistically. They proved that malwares can be detected as they will not be within the normal domain.	They just depended on the self set information. This leads to increase computational cost.
(J-Y. Xu et al., 2004)	Tracing suspected file's API sequence and using similarity measurement between two sequences.	Their model could find new strains of polymorphic infections.	They assume that the sequence of the API polymorphic malwares will not change, although the signature changed. However, there are some malwares, such as metamorphic one, will change this sequence.
(Bo-yun Zhang, 2006)	They used support vector machine to build a method to detect computer viruses.	They can minimize the dataset size with keeping detection of virus on good rate.	However, it takes more time to scan strings and hence more cost computational. The survey not included the co-stimulation process for discrimination.
(Stibor et al., 2005)	They applied two sets to the support vector machine.	They making the relation between the detectors and set size became linearly.	The detectors were not guided detectors.
(Hart and Timmis, 2008)	Did Survey on artificial immune system models	Reviewed many applications.	No touch found to co-stimulation process.
(Cheng, 2009)	They applied Byase algorithm on API sequence to detect malicious codes.	The work able to detect malwares based on known behaviors.	Conditional probability for a case obtained and compared to a threshold to decide if a case is malware or not. Some errors will records due to the threshold value.
(Miao Wang, 2009)	Tracing API using support vector machine	They included the native APIs in training set to find abnormality of a suspected file.	The wider area of API will lead to increase the false alarm of the model.
(Bradfield, 2010)	The used Formal method to define Malwares and to find their anti.	Tried to detect malwares based on the degree of correctness of a suspected file.	The method needs validation. It has been checked only on the malwares that already known. However, the correctness of unknown Malwares could not be measured.
(Khaled et al., 2010)	They used clonal selection algorithm in the work as malware classifier.	Their work can classify unknown malwares.	However, they already predefined some group of malwares and accordingly a malwares lays in one. Problems will be with malwares that have different group characteristics.
This work	Will use artificial Intelligence (ANN) or (Fuzzy logic) as a machine learning to implement immune system co-stimulation	Munising false alarm and error rate.	Biological model will mimic co-stimulation process to do confirmation so that false rate could be eliminated.

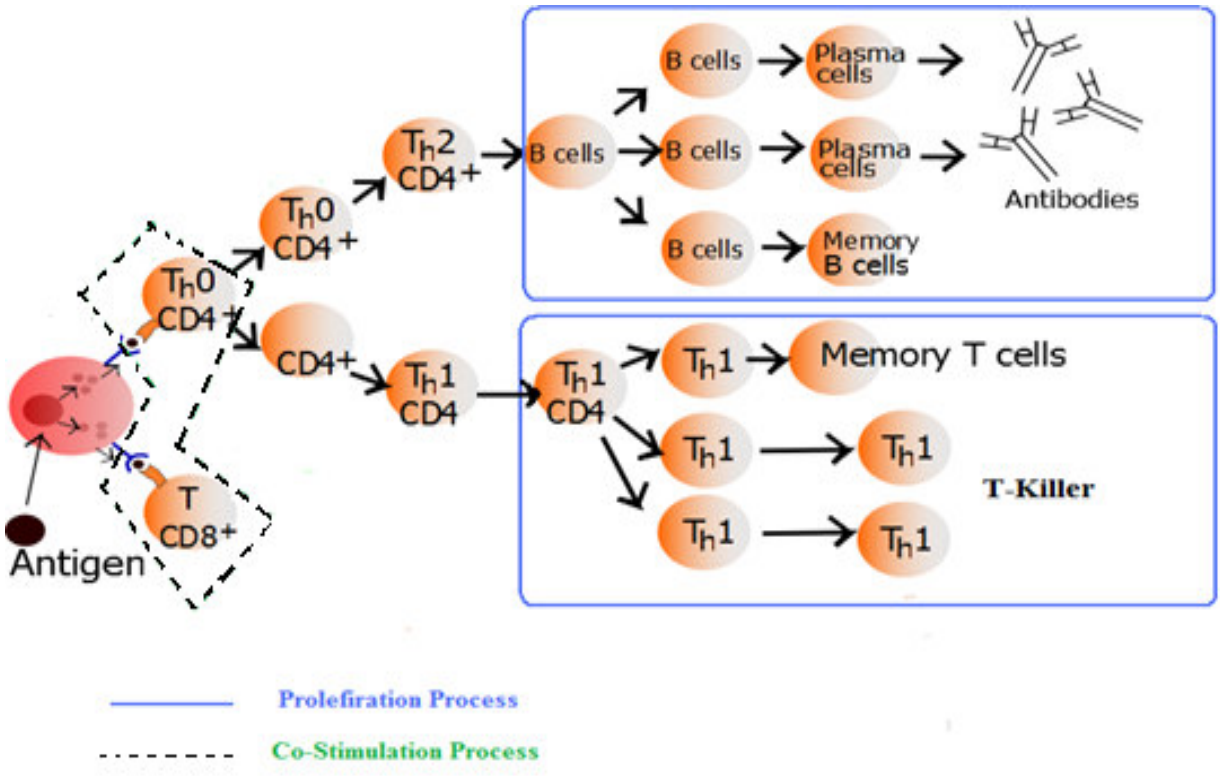

Figure 2. Process of co-stimulation and proliferation.

is a basic and essential condition for considering an antigen as a non-self cell. Without this message, the popular stage, which is the proliferation of antibodies, will not be activated. Even if activated, theoretically, this will generate improper antibodies that may possibly attack self cells. The process of self-attack means activating a self cell as an antigen (Julie et al., 2010). Such a case is similar to the process of generating a false alarm when a normal file is identified as a malware by a computer detection system (Nachenberg, 2001).

(PE) structure and infected locations

The suspected file in this work is a Windows PE file. The structure and the format of these files, as illustrated in Figure 3, play a key role in achieving the compatibility of Windows OS with its versions (Pietrek, 1994). They use the same format in linking, loading, and memory imaging of a program file executed under this environment. This also explains why virus writers spend more time in learning the PE file structure (Father, 2004).

Malware writers and detection builders should have sufficient knowledge about the structure of PE files. Malware writers use these files to hide their infection codes and payloads. Therefore, the detection system builders should know the locations of possible malware infections to modify them (Miao Wang, 2009).

Before identifying the locations and fields that malwares have probably infected, explaining the functionalities of important locations in the PE structure is more desirable.

The PE structure consists of headers and sections that explain the logical and physical information of file storage and execution. The physical part is called ‘file header”, which contains such information as number of sections and size of optional header. The logical part, known as “optional header’, has information such as “relevant virtual address, file or section alignments, address of entry points”, and many others (Wei and Ansari, 2008).

The third header, “section header”, is also called “section table”. It is a structure that contains information concerning the PE sections that follow this header. It is one of the important layers that scans for malware detection because each PE file is described in specific directory in the section header (Szor, 1998).

In general, sections are used to store data and codes of the file separately. Windows applications have nine predefined sections: .text, .bss, .rdata, .idata, .rsrc, .edata, .pdata, reloc, and .debug. Some applications may not need all of these sections, whereas others may require still more sections to suit their specific needs (Miao Wang, 2009).

Codes and instructions of the PE file are stored in the .text section, whereas data of the PE file are stored in .bss, .rdata, or .data, sections based on their types (Ye et al., 2008).

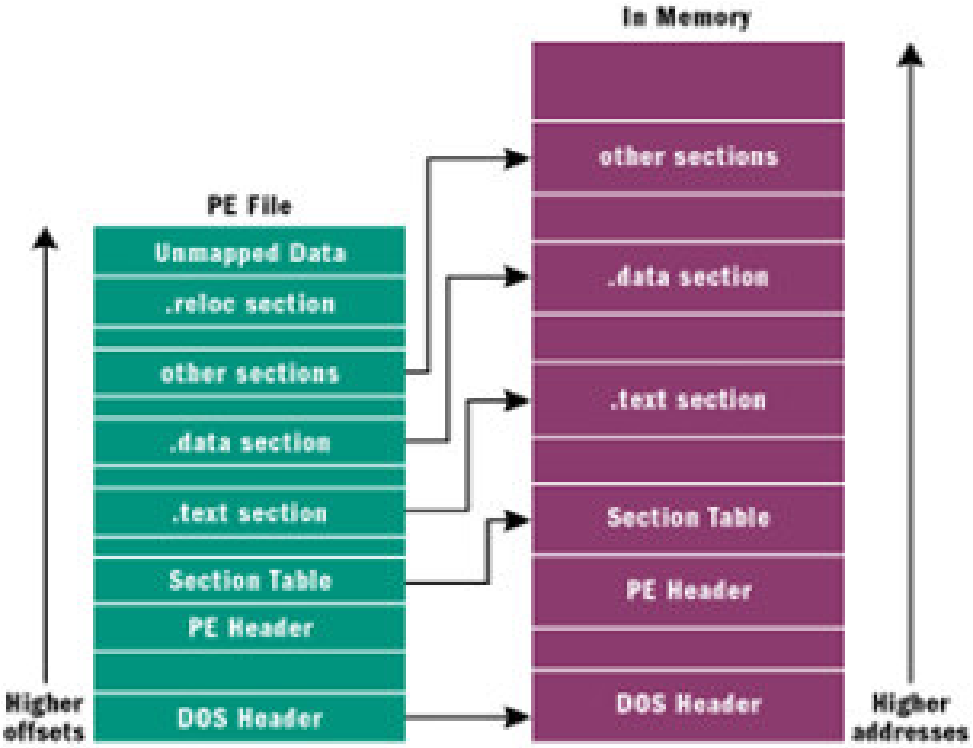

Figure 3. A typical structure of a (PE) file.

Table 2. PE locations need checking.

References	Locations scanned	Purpose of scan
(Cheng, 2009)	e-ifanew	As indicating the address where PE execution started.
(Khaled et al., 2010)	Number of sections	Malwares modify this section because mostly they add new section/s to the file.
(Cheng, 2009)	Address of entry point	Malware uses this address to point to its code address.
(Essam, 2008)	Size of image	It is indicating the memory size needed to execute the PE file. Malware will change this location.
(Essam, 2008)	Check sum	Sometime, malware modify it to indicate their presence.
(Jajodia, 2009)	Section table – size of section code (.text)	The .text section is containing codes and additional jump table for all Dll calls. Malware will insert their code in this field. The size of code increased and means changing it also in section table.

The most important sections that malwares always scan are .edata and .idata. These sections contain information about the physical addresses of the Windows functions, which are called application programmable interface (API). The .edata section contains information about APIs that the file exports, whereas .idata features information about APIs that the file imports. The “import table” in the .idata is used by malware analysts to identify whether or not a PE file is infected (Ye et al., 2008).

To build an active and efficient detection system, this paper identifies the important locations that have likely been changed by malwares. Table 2 discusses these fields and locations. This paper also explains why these locations are important and what changes malware

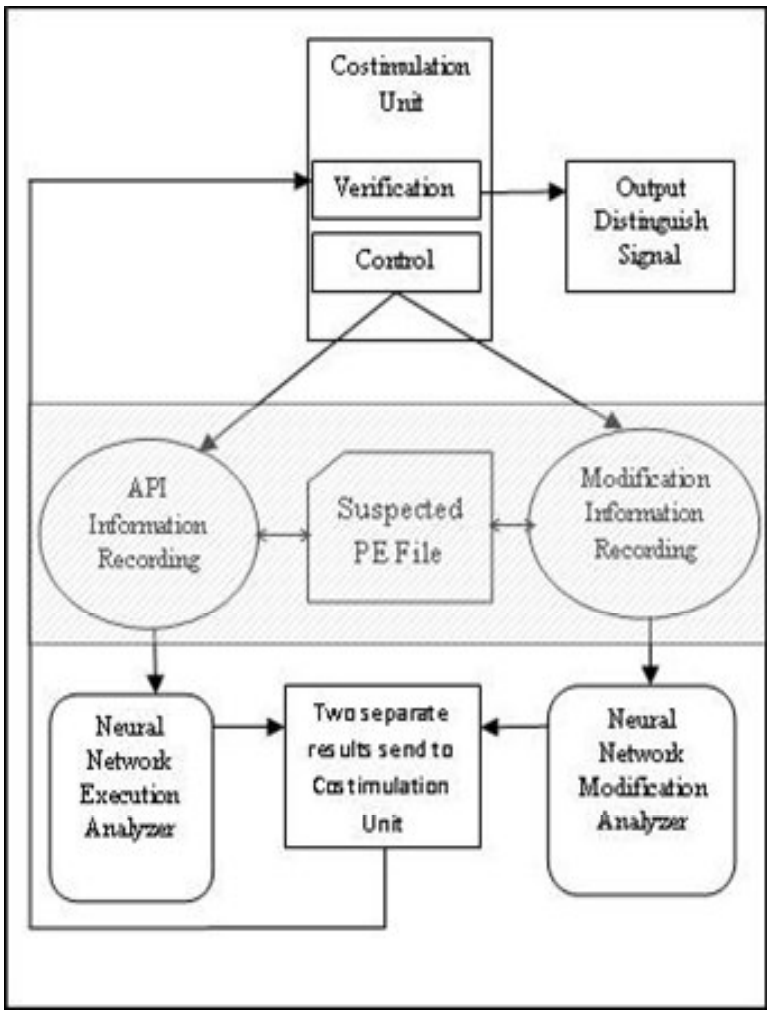

Figure 4. The (AISC) framework.

perform during infection.

One or more of the above locations and fields should be changed by malwares so that they can control the execution of PE files and execute their payloads (Clemens, 2009).

The framework of the proposed model.

Based on my best knowledge, the pervious AMI works have just depended on three types of immune system algorithms; negative selection algorithm, clonal selection algorithm and danger theory. However, the biological defense system will not activate these algorithms until the co-stimulation signals will confirm the discrimination process. The co-stimulation process will eliminate self attacking cases in the body. Self attacking process is attacking self cells and considered them as viruses. This is, somehow, like the false alarm in computer security systems when normal file detected as abnormal and vice versa.

Therefore, the inspiration of co-stimulation process in biological based detection systems could improve the accuracy of such systems.

To do that, this work proposed a model that checked fields inside the PE files that probably malware will change them during infection. Meanwhile, the execution behavior of the same file will be checked too. Both results will be compared so that the first result will confirm the second.

As illustrated in the Figure 4, the framework that suggested by this work is consist of three stages. The stages are “co-stimulation unit, scanned suspected file and prediction stage”. The part related to this work is co-stimulation unit. It performs two tasks:

Controlling: In the scanning section there are two nodes that scanned the suspected file, each in different direction; execution behavior and modification behavior. The control part of co-stimulation unit will instruct these two nodes to start scanning the identify locations in the PE file to extract

information. The information passed to stage three for analyzing.

Verification: Stage three will generate two results. These results will send as feedback to verification part of the co-stimulation unit. Depending on these results the output of the co-stimulation unit will generate and discrimination of the suspected file could confirm.

The analyzer stage will use a Feed forward back propagation neural network. The number of nodes at input layer depended on the number of features that extracted from PE file. Features will cover the modifications made on some locations by malwares and changes in execution behaviors of the infected file.

DISCUSSION

The theory that this work has just presented takes its inspiration from the co-stimulation process of human immune system. This process takes a strong and an important role to build a right decision about the normality or abnormality for a case by the immune defense system of human. It will identify the type of responses that immune system should take against unknown case and controls the respond system to not attack normal cases.
The work contributed in this review could improve the following issues in detection systems.

The model could improve the accuracy of detection systems, because it will check the suspected files with two different groups of API.

a. Improving the false alarm rate for detection systems.
b.Improving the prediction rate for unknown malwares, as the system will not depend on a threshold value or other probability application.
c. Minimizing the cost of computation for detection systems.
d. Building direct detectors that not need optimization.

There is no doubt that the above improvements will make detection systems to perform their functionalities better.

CONCLUSION

Many biological models proposed to bring solutions to computer detection systems. They are touched many algorithms such negative selection, clonal selection and danger theory. All algorithms used to build detectors and to maturate them in order to activate with a suspected case perfectly.

Most works followed with error rate in detection as they checked the suspected cases only with one dataset and sometimes with two small size dataset. Even with using two datasets, there is no verification process used between them to confirm their results.

This review work has found that there is a missed link in all proposed works to improve the detection process by decreasing the false alarm rate, which is building a confirmation system between these two datasets that extracted from the suspected case.

This work will inspired the co-stimulation process in our proposed model to perform the confirmation process, which expected to increase the accuracy of the detection processes as it does in human immune system.

REFERENCES

Aickelin U, Cayzer S (2002). The danger theory and its application to artificial immune systems, Citeseer., pp. 141-148.

Aickelin U, Greensmith J, Twycross J (2004). "Immune system approaches to intrusion detection–a review." Artificial Immune Systems, pp. 316-329.

AV-Comparative (2010). "Report Anti-Virus Comparative February 2010." 2010, from www.av-comparatives.org/images/stories/test/.../avc_report25.pdf.

Bo-yun Z, JPY, Jin-bo H, Ding-xing Z, Shu-lin W (2006). "Using Support Vector Machine to Detect Unknown Computer Viruses." Int. J. Compu. Intelli. Res., 2(1): 100-104.

Bradfield S, K a JC (2010). "A General Definition of Malware." J. Com. Virol., 6(2): 105-114.

Chappell D (2006). Understanding .NET, Addison-Wesley Professional.

Cheng WJP, Rongcai Z, Xiaoxian L(2009). Using API Sequence and Byase Algorithm to Detect Suspicious Behavior. International Conference on Communications and Networking in China, Information and Coding Theory Symposium Xi'an, China, IEEE Computer Society, pp. 26-29.

Clemens KPMC, Engin K, Xiaoyong Z, XiaoFeng W (2009). Effective and Efficient Malware Detection at the End Host. 18th USENIX Security System Montreal, Canada, USENIX Security System

Coorp S (2008). "Symantec Internet Security Threat Report Volume XIII." Whitepaper, Apr. 23.

Dasgupta D (2007). "Advances in artificial immune systems." Computational Intelligence Magazine, IEEE, 1(4): 40-49.

De Castro L, Timmis J (2002). "Artificial immune systems: a novel approach to pattern recognition" 16: 7.

D'haeseleer P, Forrest S, Helman P (2002). An immunological approach to change detection: Algorithms, analysis and implications, of IEEE Symposium on Research in Security and Privacy, Oakland, CA, pp. 110-119.

D'haeseleer P (2002). An immunological approach to change detection: Theoretical results, IEEE, pp.18-26.

Edge K, Lamont GB, Raines RA (2006). A retrovirus inspired algorithm for virus detection and optimization, ACM, pp. 103-110.

Essam AlD, Ja BZ (2008). "Computer Virus Strategies and Detection Methods." Int. J. Open Prob. Com. Sci. Math. (IJOPCM), 1(2).

Father H (2004). "Hooking Windows API - Technics of hooking API functions on Windows." COD.Break-J., 1(2).

Filiol E, Helenius M, Stefano Z (2006). "Open problems in computer virology." J. Com. Virol., 1(3): 55-66.

Forrest S, Perelson SA, Allen L (2002). Self-nonself discrimination in a computer, IEEE, pp. 202-212.

Fosnock C (2005). "Computer Worms: Past, Present, and Future." East Carolina University. 8.

Glaser B, Strauss A (2007). The discovery of grounded theory: Strategies for qualitative research, Aldine Transaction. portal.acm.org/citation.cfm?id=1864376.

Hart E, Timmis J (2008). "Application areas of AIS: The past, the present and the future." Appl. Soft Comput., 8(1): 191-201.

Health NIO (2003). "Understanding the Immuney System How it Works." Science Education. 9.

Higgins M, Ahmad K, Jacobs D, Balckburn H (2003). "Symantec Internet Security Threat Report—Attack Trends for Q3 and Q4 2002."

Symantec, Feb 2003.
Hofmery SA (2000). An interpretative introduction to immune system. Design Principles for the Immune System and Other Distributed Autonomous Systems, CiteSeer Bata., 96: 890.
Jajodia S (2009). Identifying Malicious Code Through Reverse Engineering. Advances in Information Security. A. Singh. USA, SpringerLink. 44.
Jieqiong C, Zheng Y, Wei Z (2010). "A Survey of artificial immune applications " Artificial Intelligence Review 34(1 / June, 2010): pp. 19-34.
Julie W, Greensmith A, Uwe A (2010). Artificial Immune Systems. School of Computer Science, University of Nottingham, Engineering and Physical Sciences Research Council (EPSRC), pp. 81-85.
J-Y Xu AHS, Chavez P, Mukkamala S (2004). Polymorphic Malicious Executable Scanner by API Sequence Analysis. 4th International Conference on Hybrid Intelligent Systems (HIS 2004), Kitakyushu, Japan, IEEE Computer Society 2005.
Khaled A, Abdul-Kader H, Housam R, Weiss H, Davis D, Gregory R, Gebretsadik T, Shintani A (2010). "Artificial Immune Clonal Selection Classification Algorithms for Classifying Malware and Benign Processes Using API Call Sequences." IJCSNS, 10(4): 31.
Kim J, Bentley P (2001). An evaluation of negative selection in an artificial immune system for network intrusion detection, Citeseer. GECCO, pp. 1330-1337.
Kim J, Greensmith J, Twycross J, Aickelin U (2005). Malicious code execution detection and response immune system inspired by the danger theory, Citeseer, pp. 72-75.
McAfee Internet Security (2010). McAfee Threats Report: Second Quarter 2010. n. Q. 2010: 20.
Merkel R, Hoppe T, Kraetzer C, Dittmann J (2010). Statistical Detection of Malicious PE-Executables for Fast Offline Analysis, Springer, pp. 93-105.
Miao Wang CZ (2009). Native API Based Windows Anomaly Intrusion Detection Methods Using SVM. International Conference on Senser Networks, Ubiquitous, and Trustworthy, Taichung, Taiwan, IEEE Comput. Soc., pp. 514-519.
Michael AH (1997). "Ageing, defence mechanisms and the Immune System." Age and Ageing Oxford J., 26-S24: 15-19.
Mori A (2004). Detecting Unknown Computer Viruses – A New Approach –. Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 3233/2004: 226-241.
Nachenberg CS (2001). Histogram Based Virus Detection. I. A. P. U. t. P. C. T. (PCT). Northride, CA 91326 (US).
Naik S (2003). "Introduction to The Immune System." J Indian Rheumatol Assoc 11(1, March 2003): 6.
Oney W (2002). Programming the microsoft windows driver model, Microsoft Press Redmond, WA, USA. Microsoft Press.
Paquette J (2000). "A history of viruses." Security Focus, January 16: 2004.
Piccard P (2005). Systems and Methods for Identifying Malware Distribution, US Patent App. 20,090/144,826.
Pietrek M (1994). "Peering Inside the PE: A Tour of the Win32 Portable Executable File Format." Retrieved 5/5/2010, 2010.
Sami A, Yadegari B, Hossein R (2010). Malware detection based on mining API calls, ACM, pp. 22-26.
Schmidt A, Bye R, Schmidt HG, ksel KAY, Kiraz O (2009). Static analysis of executables for collaborative malware detection on android, IEEE.
Security SE (2010). Symantec Global Internet Security Threat Report - Trends for 2009, Fond in http://securityresponse.symantec.com/business/theme.jsp?themeid=threatreport. XV.
Shevchenko A (2007). "The evolution of self-defense technologies in malware." Available from webpage: http://www. viruslist. com/analysis.
Skoudis E, Zeltser L (2004). Malware: Fighting malicious code, Prentice Hall PTR. pp. 15-24.
Smith DF (2006). The Immune System. INDIANA, USA, University of Evansville, 6: 6.
Srinivasan SRS (2009). "Intelligent agent based artificial immune system for computer security—a review " Artif. Intelleigence Rev., 32(1-4: 13-43.
Stibor T, Mohr P, Timmis J (2005). Is negative selection appropriate for anomaly detection?, ACM. GECCO, pp. 321-328.
Szor P (1998). Attacks on Win32. Virus Bulletin Conference, Munich, Germany, Virus Bull., pp. 57–84.
Ször P (2000). "Attacks On Win32–Part II." VIRUS 47.
Szor P (2006). The Art of Computer Research and Defence, Addison Wesley Profesional. pp. 02-05.
Tumer D, Entwisle S, Fossi M (2006). Symantec internet security thread report trends for January06-June06, Volume X. Symantec Inc., 2006. 9.
Turner D, Entwisle S, Fossi M (2004). "Symantec Internet security threat report." Trends for January 1.
Tyagi N, Vyas A (2008). "Data security from malicious attack: Computer Virus." 7: 11.
Wei YZZ, Ansari N (2008). "Revealing Packed Malware." Security and Privacy, IEEE, 6(5): 5.
Willems C, Holz T, Felix F (2007). "Toward automated dynamic malware analysis using cwsandbox." IEEE Security & Privacy: 32-39.
Ye Y, Wang D, Gao Y, Chen G, Gao H, Dai X (2008). "An intelligent PE-malware detection system based on association mining." J. Comput, Virol., 4(4): 323-334.
Yu Y, Hou C (2004). A clonal selection algorithm by using learning operator, pp. 26-29.
Zakaria SMAO (2009). Devising a Biological Model to Detect Polymorphic Computer Viruses Artificial Immune System (AIM): Review. 2009 International Conference on Computer Technology and Development, Kota Kinabalu, Malaysia, IEEE Computer Society.