Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2024 SCPE. Volume 25, Issues 1, pp. 45–54, DOI 10.12694/scpe.v25i1.2173

COMPUTER NETWORK VIRUS DEFENSE WITH DATA MINING-BASED ACTIVE PROTECTION

XIAOHONG LI*, YANG LI†, AND HONG HE‡

Abstract. A novel approach is presented in this paper to address the limitations of virtual machine technology, active kernel technology, heuristic killing technology, and behaviour killing technology in computer network virus defence. The proposed method provides data mining technology, specifically Object-Oriented Analysis (OOA) mining, to detect deformed and unknown viruses by analyzing the sequence of Win API calls in PE files. Experimental results showcase the Data Mining-based Antivirus (DMAV) system's superiority over existing virus scanning software in multiple aspects: higher accuracy in deformed virus detection, enhanced active defence capabilities against unknown viruses (with a recognition rate of 92%), improved efficiency, and a reduced false alarm rate for non-virus file detection. Furthermore, the paper introduces an OOA rule generator to optimize feature extraction, enhancing the system's intelligence and robustness. This research provides a promising solution to support virus detection accuracy, active defence mechanisms, and overall efficiency while minimizing false positives in virus scanning, thus contributing significantly to the advancement of computer network security.

Key words: Metamorphic virus, PE documents, Win API sequence, Data mining, OOA mining

1. Introduction. The use of the internet and the growth of online businesses have made network security a big concern. It's a problem because it threatens the safety of networks and the information stored on them. Hackers can get into systems differently, stealing user data and private information and causing systems to stop working correctly. This creates risks and challenges for businesses, governments, and individuals. Traditional security methods like operating system strengthening and firewalls are static, meaning they don't adapt well to new and complex attacks. Because of this, researchers are now working on intrusion detection systems and dynamic defence technology to better protect against online threats [11].

A typical cyber-attack can be broken down into three stages: information gathering, the actual attack, and exploiting unknown system vulnerabilities for illegal activities. Deliberate attacks often follow a specific sequence. Take Distributed Denial of Service (DDoS) attacks as an example: the process begins with identifying and scanning numerous hosts to locate a vulnerable target. Subsequently, a remote exploit program is employed to breach the target, gaining control over the system. Lastly, an attack daemon is installed and executed on the compromised host at the DDoS distribution point, which, in turn, deploys multiple compromised machines to scan and attack a single target. Studying the patterns of attack behaviours is significant in advancing intrusion detection technology [4].

Intrusion detection involves identifying unauthorized access or security breaches within a computer network or system. It gathers data from critical points in the network or system and then analyzes it to identify potential security policy violations and signs of network or system attacks. An Intrusion Detection System (IDS) is a combination of hardware and software designed for this purpose, serving as a tool to recognize potential intrusions or attacks on computer systems or networks. Intrusion detection plays a vital role within the broader security framework, serving as a dynamic defence technology that contributes significantly to overall system security [13].

The IDS can be categorized in various ways, primarily based on the source of detection data and the

*Department of Network and Communication Engineering, Shijiazhuang Information Engineering Vocational College, Shijiazhuang, Hebei, 050000, China
†Department of Network and Communication Engineering, Shijiazhuang Information Engineering Vocational College, Shijiazhuang, Hebei, 050000, China (Corresponding Author: yangli23@126.com)
‡Department of Software Engineering, Shijiazhuang Information Engineering Vocational College, Shijiazhuang, Hebei, 050000, China

intrusion detection methods employed. Regarding the source of detection data, IDS can be classified into two main types: Host-Based Intrusion Detection Systems (HIDS) and Network-Based Intrusion Detection Systems (NIDS). HIDS focuses on detecting events occurring within the host itself, such as file modifications, process creation, and system calls. In contrast, NIDS identifies events within network traffic, including network connections and packet content [1].

The authors used another classification technique based on the intrusion detection methods. This categorization divides IDS into two distinct approaches: misuse detection and anomaly detection. Misuse detection identifies known attack patterns through predefined rules, typically created by security experts. On the other hand, anomaly detection relies on learning algorithms to identify unknown attack behaviours, often employing machine learning and data mining techniques to discover abnormal activities [23].

IDS functions as a crucial second line of defence behind firewalls, offering real-time intrusion detection and implementing appropriate security measures such as evidence collection for tracking, network connection termination, and recovery. However, as intrusion detection technology evolves, so does intrusion technology. The demand for security products based on multi-level and comprehensive defence strategies has emerged. The evolution of intrusion detection systems has progressed through three stages: from active response within the intrusion detection system to interaction between the intrusion detection system and the firewall, and ultimately to the latest development of intrusion prevention systems [14].

The current state of intrusion prevention systems can be viewed as a fusion of firewall and intrusion detection capabilities. These systems leverage intrusion detection as their basis for identifying potential threats, but they take proactive measures to thwart attacks once detected. While introducing intrusion prevention systems has partially met the demands of interconnected network users within specific environments, they still exhibit shortcomings, including adaptability, scalability, limited intrusion event analysis capabilities, reliance on a single defence approach, and difficulties in handling complex attacks [17].

Intrusion prevention systems face challenges due to the constantly changing network environment and evolving attack techniques, which hinder their ability to stay up-to-date with emerging attack methods and malicious behaviours. This vulnerability renders them susceptible to evasion by attackers and reduces their overall effectiveness. Additionally, their rigid design and implementation constrain their adaptability and customization to diverse network setups and attack scenarios, limiting their practicality and scalability. Furthermore, these systems predominantly focus on detecting and reporting intrusion events, neglecting comprehensive analysis and traceability, which in turn hampers their ability to facilitate timely responses and efficiently manage intrusion incidents. Addressing these deficiencies in intrusion prevention systems is essential to strengthen their resilience and enhance network security in the face of evolving threats and complex attack landscapes [7].

To investigate security systems founded on distributed technology, featuring adaptable and open structures and incorporating the seamless integration of intrusion detection and defence technologies are proposed. Such research and development efforts are essential for advancing network information security technology and products towards comprehensive three-dimensional protection and bolstering national security defence initiatives [9].

The paper is structured into five sections with an overview of the research topic and outlines the study's objectives. Section 2 offers a comprehensive literature review, presenting the existing knowledge and research in computer network virus defence, including various technologies and approaches. The details of the proposed method, explaining the utilization of data mining technology for virus defence and the specific techniques, are employed in Section 3. The results of experiments and subsequent discussions and the performance of the proposed method are discussed in Section 4. Finally, Section 5 provides a conclusion summarizing the key findings and contributions.

2. Literature Review. Currently, the predominant antivirus technology still relies on signature-based methods, and its primary drawback lies in the static nature of the signature codebase, necessitating continuous updates to combat emerging viruses. This signature-based approach often results in a reactive defence stance, where antivirus solutions lag behind new threats. Researchers have introduced novel antivirus technologies to tackle this challenge, including behaviour detection, machine learning, and sandboxing. These approaches avoid reliance on fixed signature codes, instead analyzing virus samples' behaviours and characteristics to determine their malicious nature. In contrast to signature-based techniques, these innovative technologies exhibit superior adaptability and generalization capabilities, enabling them to counteract new and mutated

viruses effectively. Consequently, the evolution towards intelligent virus-active defence systems represents an inevitable virus detection and mitigation progression [21].

Prominent domestic antivirus software companies like Kingsoft, Rising, and Jiangmin primarily rely on traditional signature scanning technology for virus detection and eradication. However, they also commit to integrating advanced technologies into their antivirus strategies. For instance, Jinshan Antivirus Laboratory has harnessed the power of suspicious tool scanning and "honeypot" technology, resulting in notable successes in identifying suspicious files. Their in-memory virus detection technology effectively identifies and eliminates highly concealed viruses. Ruixing, on the other hand, has ventured into semi-virtual machine technology as a means to detect previously unknown viruses. Jiangmin, meanwhile, has implemented the "signature broad-spectrum method" to detect and thwart certain forms of deformed viruses. This diversity of approaches, in line with the "hundred schools of thought contend, each with its own merits", collectively furnishes effective security assurances for the computer systems of most users [18].

The authors introduced an innovative Android virus software detection method that combines the strengths of learning-based and signature-based approaches. This method automatically synthesizes semantic malware signatures from a limited number of instances within a virus software family. The common functionalities shared by the virus software family are captured by the pattern represented through the call graph between components of Android applications. Utilizing MaxSAT, the virus software signature is synthesized by detecting the Maximum Suspicious Common Subgraph (MSCS) within a group of ICCG [2].

Optimized network structures are explained using genetic algorithms, comparing the performance with traditional Back-Propagation (BP) and Levenberg–Marquardt (LM) algorithms. Their comparison reveals that the GA-LM model accurately predicts Dissolved Oxygen (DO) values and outperforms traditional neural networks as a rapid interpolation and extrapolation tool [16].

The authors introduced executable malware detection technology, leveraging cluster analysis techniques to categorize executable files into multiple feature regions. This approach uses cluster analysis to detect known and unknown malware and measures byte distribution similarity between malicious and normal executable files. Consequently, this technology efficiently identifies malicious software without complex command analysis, minimizing system execution overhead [15].

The contributors introduced an active virus defence approach that employs object-oriented association mining technology within the Windows platform. By conducting a static analysis of WinAPI call sequences within PE files and integrating it with OOA mining technology, authors have developed and executed a DMAV system [22].

3. Research Methods. The primary structure of the DMAV system is illustrated in Figure 3.1. Initially, when dealing with potentially suspicious PE files that have been compressed (e.g., UPX, ASPack, etc.), the PE file parser extracts all WinAPI call sequences from the imported table, following the order of code execution. Each function within the sequence is assigned a unique 32-bit integer ID through a database query based on the API calls. This research employs the same methodology to derive API functions called by all samples within the training dataset. Subsequently, the WinAPI call sequences extracted from all samples are stored as integer vectors in the feature database. To actively defend against polymorphic and unknown viruses, this study incorporates the OOA mining algorithm from data mining technology to analyze the feature database.

Through a rule generator, it identifies and stores association rules that fulfil specific criteria in a rule database. To determine whether a suspicious PE file is a virus, the system compares the WinAPI sequence it generates with the antecedents of each rule in the rule library. The suspicious PE file is categorized as a virus if the distance between vector V_u and V_s surpasses a predefined threshold [6]. This approach offers the advantage of rapid scanning and detecting numerous PE files while maintaining a high detection rate for known viruses.

3.1. PE file parser. The Portable Executable (PE) file format is the predominant executable file format within the current Windows platform. In this paper, we have developed a PE file analyzer designed to extract all WinAPI call sequences found in the PE file's import table. The implementation of this parser is structured into three key steps:

Step 1: The import table is initially located within the PE file, following the PE file structure. This table contains pointers to the serial numbers and names of all WinAPI input functions.

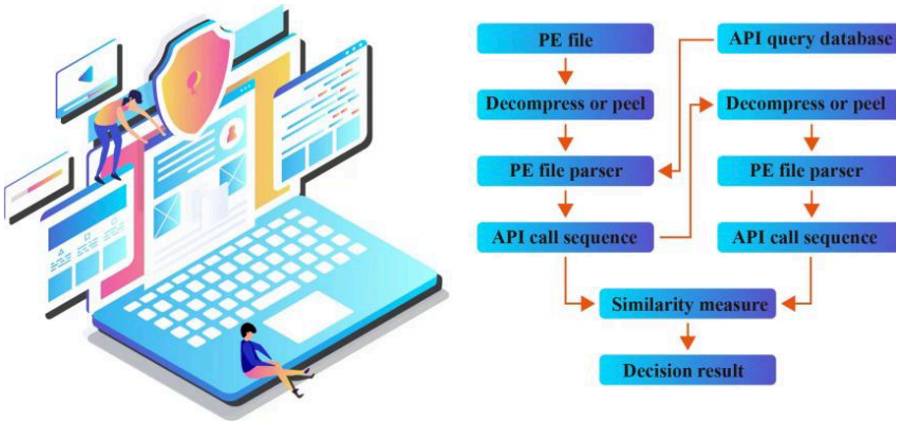

Fig. 3.1: Main structure of DMAV system

Table 3.1: Sample file database

ID	API sequence	Sample category
1	API1,API5	BENIGN
2	API1,API3	TROJAN
3	API1,API2,API4,API5	BENIGN
4	API1,API2,API3,API4,API5	WORM
5	API3,API5	BACKDOOR
6	API2,API4	BENIGN
7	API1,API4,API5	SPYWARE
8	API2,API5	BENIGN

Step 2: Next, all code segments are systematically scanned within the PE file, extracting the corresponding call instructions and their associated target addresses based on the sequence of code execution. Subsequently, the corresponding WinAPI input functions are identified by cross-referencing the target addresses with the input address table.

Step 3: Using the API query database, each exported WinAPI function is mapped to a globally unique 32-bit integer API ID number to streamline the process. This database stores the names and ID numbers of commonly used Win32DLLs and their API functions. By representing the WinAPI sequence as a 32-bit integer vector, we significantly enhance the efficiency of similarity measurement, saving valuable computation time [12, 8].

3.2. Rule generator. Both decision trees and Bayesian networks have been employed in virus detection, but these methods are susceptible to a common issue known as over-learning. Over-learning occurs when a model performs well on training data but poorly on test data due to excessive complexity or inadequate training data. Excessive model complexity can lead to memorization of the training data's characteristics without the ability to generalize to new test data. In contrast, insufficient training data prevents the model from acquiring enough information for effective learning.

This paper integrates the OOA mining algorithm from data mining technology to address these challenges. It analyses 5000 viruses and 2000 non-virus samples using a PE file parser, storing their distinctive attributes in a database, as shown in Table 3.1. Subsequently, it mines association rules tailored to specific objectives through a rule generator and archives them in a rule database. This paper optimizes feature extraction processes to enhance system efficiency further [5, 10].

In the context of association rule mining, the OOA mining algorithm distinguishes itself from constraint-

based association mining by focusing on extracting association rules that fulfil specific objectives and demonstrate utility. The rule extraction process employed in this paper exemplifies an application of OOA mining. As illustrated in Table 3.1, we are engaged in mining association rules that satisfy the target criteria Obj = (Group = MALICIOUS (TROJAN ∨ WORM ∨ BACKDOOR ∨ SPYWARE)), essentially aiming to differentiate Obj = (Group = ¬ BENIGN).

3.2.1. Related concepts. Set the database DB, and the item set $I = \{i_1, i_2, \ldots, i_m\}$ is composed of two parts: $I = I_{\text{obj}} \cup I_{\text{nobj}}, I_{\text{obj}} \cap I_{\text{nobj}} = \emptyset, I_{\text{obj}}$ is the item set that constitutes the target, I_{nobj} is the antecedent that constitutes the association rule to be mined, and the transaction set T has n transactions, specifying the minimum target support mos% and the minimum target confidence moc%.

Definition 3.1. *Objective (Obj) is a logical formula composed of items in* I_{obj}. $I_{\text{obj}} = \{Group\}, \text{Obj} = (Group = \neg\text{BENIGN})$ *can be set. If* Obj *is true in a transaction* $t \in T, t$ *is said to meet the goal. If there is* $X \subseteq t\,(X \subseteq I_{nobj})$ *at the same time, then* $X \rightarrow$ Obj *is satisfied in* t. *OOA mining is to mine rules like* $X \rightarrow$ Obj.

Definition 3.2. *If project set* $X = \{a_1, a_2, \cdots, a_i\}$ *is set, and* $X \subseteq I_{nobj}$, *the transaction count that meets* X *is count* (X, DB), *and the transaction count that meets* $X \rightarrow Obj$ *is count* $(X \rightarrow \{Obj\}, DB)$, *then the target support os% and target confidence oc% of* $(X \rightarrow \text{Obj})$ *are followed using Equation* (3.1)*:*

$$\begin{aligned} os\ \% &= \frac{\text{count}(X \cup \{Obj\}, DB)}{|DB|} \times 100\% \\ oc\ \% &= \frac{\text{count}(X \cup \{Obj\}, DB)}{\text{count}(X, DB)} \times 100\% \end{aligned} \tag{3.1}$$

If os% $\geq$ mos %, X *is considered as frequent* OOA.

Definition 3.3. *If the project set* X *has* os% $\geq$ mos % *and* oc% $\geq$ moc %, *then* $X \rightarrow$ Obj(os%, *oc%) is an OOA rule.*

Theorem 3.4. *If the item set* X *is OOA frequent,* $\forall Y \subset X, Y \neq \emptyset$, *then* Y *is also OOA frequent.*

Theorem 3.5. *If the item set* X *is not* OOA *frequent,* $\forall Y \supset X, Y \subseteq$, *then* Y *is also* OOA *frequent.*

3.2.2. Main algorithm implementation of rule generator. OOA mining meets the two basic properties of the Apriori algorithm, so OOA mining can be implemented with the Apriori algorithm.

Taking Table 3.1 as an example, if mos%=25%, moc%=65%, then the frequent set FP={API1}, {API2}, {API3}, {API4}, {API5}, {API1, API3}, {API2, API4}, {API2, API5}, {API3, API5}, {API4, API5}, {API2, API4, API5}. The rules obtained are: 1) {API3} → Obj (37.5%, 100%); 2) {API1,API3} → Obj(25%,100%); 3) {API3,API5} → Obj(25%,100%); 4) {API4,API5} → Obj(25%,66.7%); 5) {API2,API4,API5} → Obj(25%,66.7%).

3.3. Similarity measurement. The similarity measurement is implemented in two steps, namely, sequence rearrangement and similarity calculation.

3.3.1. Sequence rearrangement. A sequence rearrangement procedure is employed based on two feature vectors to enhance the precision of similarity measurement between the two vectors before performing the similarity calculation. This sequence rearrangement algorithm can be executed using a matrix, as depicted in Figure 3.2.

The specific implementation of sequence rearrangement is as follows: If the first row and first column of matrix $A_{(M+1)*(N+1)}$ are row 0 and column 0, then a_{ij} can be obtained using Equation (3.2),

$$\begin{aligned} a_{ij} = \begin{cases} 0, \text{ If } a_i 0 \neq a_{a_j} \\ 1, \text{ If } a_i 0 = a_{a_j} \end{cases}, (i \in [1, M], j \in [1, N], \\ M = \text{ length(Sequence1)}, N = \text{ length(Sequence2))} \end{aligned} \tag{3.2}$$

Let's explore an illustrative instance of the sequence rearrangement algorithm to provide a more concrete demonstration. Following the meticulous sequence rearrangement procedure outlined in Figure 3.3, this paper intentionally inserts the value 0 into any vacant positions within the sequence. This deliberate step gives rise to the creation of two fresh API sequences, denoted as $BV_s{}'$ and $V_u{}'$.

	W	A	N	D	D	R
W	X					
A		X				
R				X		
E					X	
R						X
S						

Fig. 3.2: Sequence rearrangement algorithm represented by matrix

	W	A	N	O	E	R
W	1	0	0	0	0	0
A						
D						
E						
R						
S						

	W	A	N	O	E	R
W	1	0	0	0	0	0
A	0	2	1	1	1	1
D						
E						
R						
S						

	W	A	N	O	E	R
W	1	0	0	0	0	0
A	0	2	1	1	1	1
D	0	1	2	3	2	2
E						
R						
S						

	W	A	N	O	E	R
W	1	0	0	0	0	0
A	0	2	1	1	1	1
D	0	1	2	3	2	2
E	0	1	2	2	4	3
R						
S						

	W	A	N	O	E	R
W	1	0	0	0	0	0
A	0	2	1	1	1	1
D	0	1	2	3	2	2
E	0	1	2	2	4	3
R	0	1	2	2	3	5
S						

	W	A	N	O	E	R
W	1	0	0	0	0	0
A	0	2	1	1	1	1
D	0	1	2	3	2	2
E	0	1	2	2	4	3
R	0	1	2	2	3	5
S	0	1	2	2	3	4

Fig. 3.3: Example of sequence rearrangement algorithm

3.3.2. Similarity calculation. The similarity is calculated by taking the mean value of Equations (3.4) to (3.6). The most commonly used method is Euclidean distance method to calculate the similarity between two vectors, as given by Equation (3.3). However, Euclidean distance cannot accurately measure the similarity between two vectors. Equation (3.4) calculates the cosine of the angle between two vectors $V_s{}'$ and $V_u{}'$ to determine their similarity, Equation (3.5) is the Jaccard extended cosine algorithm, and Equation (3.6) is the

Table 4.1: Identifying deformed viruses using various virus-scanning software

Samples	N	M	D	K	SAVE	DMAV
Beagl	✓	✓	✓	✓	✓	✓
Beagl1	✓	✓	×	✓	✓	✓
Beagle2	✓	×	×	✓	✓	✓
Beagle3	×	×	×	✓	×	✓
Beagle4	✓	✓	×	×	×	✓
Blaster	✓	✓	✓	✓	✓	✓
Blaster1	✓	✓	✓	✓	✓	✓
Blaster2	×	×	×	×	×	✓
Lovedoor	✓	✓	✓	✓	✓	✓
Lovedoor1	×	×	✓		✓	✓
Lovedoor2	×	×	×	×	×	✓
Lovedoor3	×	✓		✓	×	✓
Mydoom	✓	✓	✓	✓	✓	✓
Mydoom1	×	×	×	×	×	✓
Mydoom2	×	×	×	?	✓	✓

Pearson correlation measurement algorithm [20].

$$D\left(V_s', V_u'\right) = \sum_{i=1}^{\min\left(\left|V_s'\right|,\left|V_u'\right|\right)} \left[\left(V_{s_i}' - V_{u_i}'\right)^2\right]^{1/2} \tag{3.3}$$

$$s^{(C)}\left(V_s', V_u'\right) = \frac{V_s'^T V_u'}{\left\|V_s'\right\|_2 \cdot \left\|V_u'\right\|_2}, \|v\|_p = \left[\sum_{i=1} \mid V_i\mid^7\right]^{1/p} \tag{3.4}$$

$$S^{(J)}\left(V_s', V_u'\right) = \frac{V_s'^T V_u'}{\left\|V_s'\right\|_2^2 + \left\|V_u'\right\|_2^2 - V_s'^T V_u'} \tag{3.5}$$

$$s^{(P)}\left(V_s', V_u'\right) = \left[\frac{\left(V_s' - \nabla_s'\right)^T \left(V_u' - \nabla_u'\right)}{\left\|\left(V_s' - \nabla_s'\right)\right\|_2 \cdot \left\|\left(\left(V_u' - \nabla_u'\right)\right)\right\|_2} + 1\right] /2 \tag{3.6}$$

4. Result Analysis and Discussion. The DMAV system is developed within the VC++6.0 environment, coupled with a MySQL database. VC++6.0 is an integrated development environment well-suited for C++ development on the Windows platform. It offers robust tools and libraries that expedite the creation of Windows applications, including virus detection systems. Utilizing VC++6.0 streamlines the development of Windows GUI applications, and frameworks like MFC can be employed to simplify the development process further. The experiment used 5,000 virus samples and 2,000 non-virus samples for rule extraction, while 150 virus samples and 500 non-virus samples served as test specimens. All samples were sourced from the Jinshan Antivirus Laboratory. The experiment is primarily divided into two components: detecting deformed viruses and detecting unknown viruses.

4.1. Detection of deformation virus. The experiment and analysis involved Win32PE virus samples, including well-known examples such as Lovedoor, Mydoom, Blaster, and Beagle. Virus deformation technology is employed for each type of virus sample to transform these samples into various deformed versions. Subsequently, different virus scanning software and the DMAV system were used to scan and assess these deformed viruses. The experiment outcomes are summarized in Table 4.1, revealing that the DMAV system consistently exhibits superior accuracy in detecting deformed viruses compared to other conventional virus scanning software.

Table 4.2: Detection of unknown viruses through different virus-scanning software

Malware Samples	N	M	D	K	SAVE	DMAV
1	✓	✓	✓	✓	✓	✓
2	✓	✓	✓	✓	✓	✓
3	✓	✓	×	✓	✓	✓
4	✓	×	×	×	×	×
5	×	✓	✓	×	✓	✓
6	✓	✓	✓	✓	✓	✓
7	✓	✓	✓	✓	✓	✓
8	×	×	×	×	✓	✓
9	✓	✓	✓	✓	×	✓
10	?	✓	×	✓	✓	×
... ...	... ...	... ...	... ...	... ...	... ...	... ...
150	✓	×	×	×	✓	✓
Statistics	50	68	48	75	82	138
Ratio/%	33.4	453	32	50	54.8	92

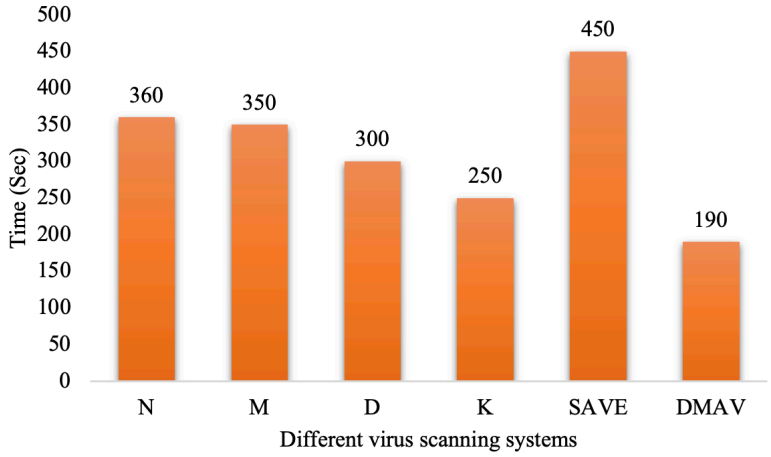

Fig. 4.1: Comparison of the efficiency of different virus scanning systems

4.2. Detection of unknown virus. To evaluate the detection capabilities of the DMAV system concerning unknown viruses, 150 virus samples with undisclosed characteristics were subjected to scanning using various virus scanning software and the DMAV system. The outcomes of this experiment are presented in Table 4.2, highlighting that the DMAV system demonstrates notably heightened efficiency in actively defending against unknown viruses compared to other conventional antivirus software. Impressively, the DMAV system achieves a recognition rate of 92%.

Remarks: ✓indicates successful detection, × indicates failed detection, and ? indicates suspicious. All scanning software adopts the latest version. N-Norton, M-McAfee, D-DF Web, K-Kaspersky, SAVE-Static Analyzer for Vicious Executable, DMAV-Data Mining-based, Antivirus system.

To assess the system's efficiency and false positive rate (FP), we conducted experiments within the same testing environment as the SAVE system. The SAVE system is an open-source network security monitoring and defence system with many functions, including intrusion detection, vulnerability scanning, and traffic monitoring. The SAVE system incorporates various technologies, such as rule-based detection, anomaly-based detection, and machine learning, to enhance detection efficiency and diminish false positives.

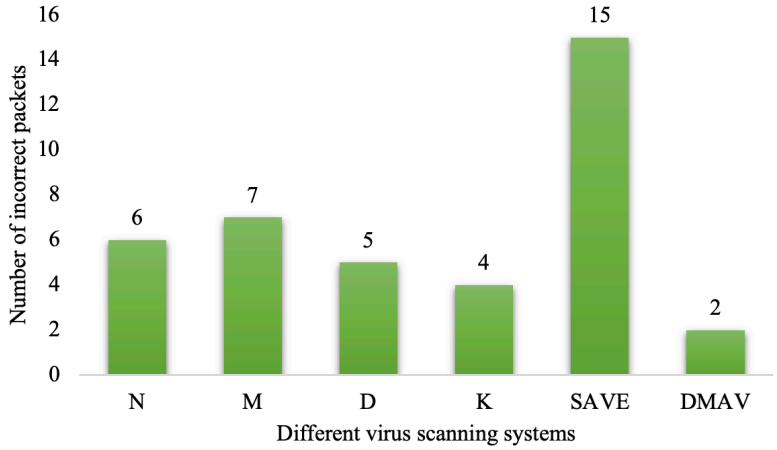

Fig. 4.2: Comparison of false alarm rates of different virus scanning systems

The proposed system is evaluated using the test environment consisting of an Intel P4 1GHz processor with 1GB of RAM, running on MS Win2000. The experimental outcomes, illustrated in Figures 4.1 and 4.2, underscore that the DMAV system exceeds other virus scanning software's efficiency and capacity to maintain a relatively lower false positive rate when detecting non-virus files than traditional virus scanning software [3, 19].

5. Conclusion. By employing static PE file analysis to extract WinAPI call sequences, a proactive virus defence system named DMAV has been proposed, operating within the Windows platform and rooted in data mining technology. DMAV performs the dual function of detecting deformed viruses and actively safeguarding against unknown ones. This system consists of three fundamental modules: a PE file parser, a rule generator, and a similarity measurement algorithm. The rule generator, empowered by OOA mining, streamlines feature extraction processes. Empirical findings demonstrate that DMAV exceeds conventional virus scanning software in terms of accuracy, efficiency, and false alarm rate when detecting unknown and deformed viruses. The computer virus defence domain, propelled by data mining, is dynamic and cutting-edge. As computer networks continue to evolve, traditional antivirus methodologies prove inadequate, prompting researchers to integrate data mining into virus defence strategies seamlessly. The forthcoming developments in data mining-driven virus defence may encompass the adoption of deep learning for more precise models and the utilization of multimodal data analysis techniques to enhance accuracy through the joint analysis of various data types, including text, images, videos, and audio, thereby stimulating network security in the face of ever-evolving virus threats.

REFERENCES

[1] A. T. Azar, E. Shehab, A. M. Mattar, I. A. Hameed, and S. A. Elsaid, *Deep learning based hybrid intrusion detection systems to protect satellite networks*, Journal of Network and Systems Management, 31 (2023), p. 82.
[2] M. Campion, M. Dalla Preda, and R. Giacobazzi, *Learning metamorphic malware signatures from samples*, Journal of Computer Virology and Hacking Techniques, 17 (2021), pp. 1–17.
[3] D. Dasgupta, Z. Akhtar, and S. Sen, *Machine learning in cybersecurity: a comprehensive survey*, The Journal of Defense Modeling and Simulation, 19 (2022), pp. 57–106.
[4] A. B. de Neira, B. Kantarci, and M. Nogueira, *Distributed denial of service attack prediction: Challenges, open issues and opportunities*, Computer Networks, 222 (2023), p. 109553.
[5] L. Demetrio, B. Biggio, G. Lagorio, F. Roli, and A. Armando, *Functionality-preserving black-box optimization of adversarial windows malware*, IEEE Transactions on Information Forensics and Security, 16 (2021), pp. 3469–3478.
[6] M. Gharacheh, V. Derhami, S. Hashemi, and S. M. H. Fard, *Proposing an hmm-based approach to detect metamorphic malware*, in Proceedings of the 4th Iranian Joint Congress on Fuzzy and Intelligent Systems, Zahedan, Iran, 2015, IEEE, pp. 1–5.
[7] E. Gyamfi and A. Jurcut, *Intrusion detection in internet of things systems: a review on design approaches leveraging multi-access edge computing, machine learning, and datasets*, Sensors, 22 (2022), p. 3744.

[8] X. Huang, L. Ma, W. Yang, and Y. Zhong, *A method for windows malware detection based on deep learning*, Journal of Signal Processing Systems, 93 (2021), pp. 265–273.
[9] A. K. Jha, A. Vaish, and S. Patil, *A novel framework for metamorphic malware detection*, SN Computer Science, 4 (2022), p. 10.
[10] D.-y. Kim, A.-y. Jeong, and T.-j. Lee, *Analysis of malware group classification with explainable artificial intelligence*, Journal of the Korea Institute of Information Security & Cryptology, 31 (2021), pp. 559–571.
[11] Q. Li, J. Hou, S. Meng, and H. Long, *GLIDE: a game theory and data-driven mimicking linkage intrusion detection for edge computing networks*, Complexity, 2020 (2020), pp. 1–18.
[12] Y. T. Ling, N. F. M. Sani, M. T. Abdullah, and N. A. W. A. Hamid, *Metamorphic malware detection using structural features and nonnegative matrix factorization with hidden markov model*, Journal of Computer Virology and Hacking Techniques, 18 (2021), pp. 1–21.
[13] Z. Lv, D. Chen, B. Cao, H. Song, and H. Lv, *Secure deep learning in defense in deep-learning-as-a-service computing systems in digital twins*, IEEE Transactions on Computers, (2023).
[14] A. Madhuri, V. E. Jyothi, S. P. Praveen, S. Sindhura, V. S. Srinivas, and D. L. S. Kumar, *A new multi-level semi-supervised learning approach for network intrusion detection system based on the 'goa'*, Journal of Interconnection Networks, (2022), p. 2143047.
[15] R. Ognev, E. Zhukovskii, and D. P. Zegzhda, *Detection of malicious executable files based on clustering of activities*, Automatic Control and Computer Sciences, 55 (2021), pp. 1092–1098.
[16] A. A. Ojugo, C. O. Obruche, and A. O. Eboka, *Quest for convergence solution using hybrid genetic algorithm trained neural network model for metamorphic malware detection*, ARRUS Journal of Engineering and Technology, 2 (2022), pp. 12–23.
[17] T. Saba, A. Rehman, T. Sadad, H. Kolivand, and S. A. Bahaj, *Anomaly-based intrusion detection system for iot networks through deep learning model*, Computers and Electrical Engineering, 99 (2022), p. 107810.
[18] V. F. Santos, C. Albuquerque, D. Passos, S. E. Quincozes, and D. Mossé, *Assessing machine learning techniques for intrusion detection in cyber-physical systems*, Energies, 16 (2023), p. 6058.
[19] S. M. Shareef and S. H. Hashim, *Proposed hybrid classifier to improve network intrusion detection system using data mining techniques*, Engineering and Technology Journal, 38 (2020), pp. 6–14.
[20] D. Shin and J. Shim, *A systematic review on data mining for mathematics and science education*, International Journal of Science and Mathematics Education, 19 (2021), pp. 639–659.
[21] M. U. Ullah, A. Hassan, M. Asif, M. Farooq, and M. Saleem, *Intelligent intrusion detection system for apache web server empowered with machine learning approaches*, International Journal of Computational and Innovative Sciences, 1 (2022), pp. 21–27.
[22] C. Xiong, Z. Li, Y. Chen, T. Zhu, J. Wang, H. Yang, and W. Ruan, *Generic, efficient, and effective deobfuscation and semantic-aware attack detection for powershell scripts*, Frontiers of Information Technology & Electronic Engineering, 23 (2022), pp. 361–381.
[23] H. Xu, Z. Sun, Y. Cao, and H. Bilal, *A data-driven approach for intrusion and anomaly detection using automated machine learning for the internet of things*, Soft Computing, (2023), pp. 1–13.

Edited by: Venkatesan C
Special issue on: Next Generation Pervasive Reconfigurable Computing for High Performance Real Time Applications
Received: Mar 21, 2023
Accepted: Sep 30, 2023