Comprehensive approach to the detection and analysis of polymorphic malware

Maksym Chaikovskyi[1,*,†], Inna Chaikovska[1,†], Tomas Sochor[2,†], Inna Martyniuk [1,†] and Oleksii Lyhun[1,†]

[1] *Khmelnytskyi National University, Instytuts'ka Str. 11, 29000, Khmelnytskyi, Ukraine*
[2] *Prigo University, Havirov, Czech Republic*

Abstract

The article examines the features of modern polymorphic malware and its impact on the functioning of computer systems. Existing approaches and methods of its detection and analysis are considered, such as: string search algorithm, intelligent data analysis, sandbox analysis, machine learning, structural feature engineering. Their advantages and disadvantages are determined. The necessity of using a new approach, namely the detection of malicious software using probabilistic logical networks, is argued. Its advantages and development prospects are determined. In the study, a comprehensive approach consisting of 3 stages is proposed for the detection of polymorphic malware. The first one uses string search algorithms. The second is a complex of methods, including intelligent data analysis, sandbox analysis, machine learning, and structural feature engineering. In the third step, the use of probabilistic logical networks is proposed, which will allow establishing the probability that the software belongs to polymorphic malware. The use of the proposed integrated approach will also allow to determine the necessary methods for neutralization of detected malicious software. This approach will maximize the probability of detecting polymorphic malware.

Keywords

malicious software, string search algorithm, intelligent data analysis, sandbox analysis, machine learning, structural feature engineering, probabilistic logic networks, complex approach

1. Introduction

The search for and elimination of computer viruses is becoming an increasingly urgent and complex problem every year. After all, they pose a threat to the smooth functioning of computer systems that are used in increasingly critical areas of human activity. Therefore, the development of methods and means of neutralizing malicious software is one of the promising and priority research tasks in the field of computer science. Despite the continuous improvement of anti-virus software, the generation and distribution of malicious software increases year by year. One of the most serious problems faced by the developers of antivirus

ICyberPhyS-2024: 1st International Workshop on Intelligent & CyberPhysical Systems, June 28, 2024, Khmelnytskyi, Ukraine
* Corresponding author.
† These authors contributed equally.
max.chaikovskyi@gmail.com (M. Chaikovskyi); inna.chaikovska@gmail.com (I. Chaikovska); tomas.sochor@osu.cz (T. Sochor); inmartunyk@ukr.net (I. Martyniuk); oleksii.lyhun@gmail.com (O. Lyhun)
0000-0002-9596-6697 (M. Chaikovskyi); 0000-0001-7482-1010 (I. Chaikovska); 0000-0002-1704-1883 (T. Sochor); 0009-0007-7751-8974 (I. Martyniuk); 0009-0004-5727-5096 (O. Lyhun)

software is the automatic mutation of the code of the malicious program. The mechanism of mutation and permutation of malicious program code is called polymorphism. Polymorphic malware cannot be identified by signature analysis. Therefore, for this purpose, it is necessary to use new, improved methods of analysis of modern malicious soft.

2. Literature Review

Among the scientists who studied the issue of detection and analysis of malicious software, the following can be distinguished: O. Savenko [1-4], S. Lysenko [1-4], A. Nicheporuk [1-4], A. Damodaran [6], K. Brezinski [8], M. Singh [9], B. Anderson [10], L. Bilge [11], U. Urooj [12], K. Gundogan [13] etc.

Among the latest methods of analysis of modern malware [1-5] are some artificial intelligence (machine learning) algorithms that analyze a malicious program in a virtual machine. A virtual machine can run a packaged potentially dangerous file and dynamically analyze it, automatically testing code and behavior. In addition, the latest research looks promising, where anti-virus software uses modern machine learning methods and real-time behavior analysis in combination with static methods to identify suspicious activity and prevent threats. This approach to malware detection is called hybrid [6]. The importance and relevance of the topic of protection against malicious software is also evidenced by statistical data. Thus, according to the statistical company Statistica, the number of cyber attacks on computer systems is constantly increasing from year to year, which is shown in (Figure 1), and the number of attacks on computer systems by types of malicious software in Figure 2.

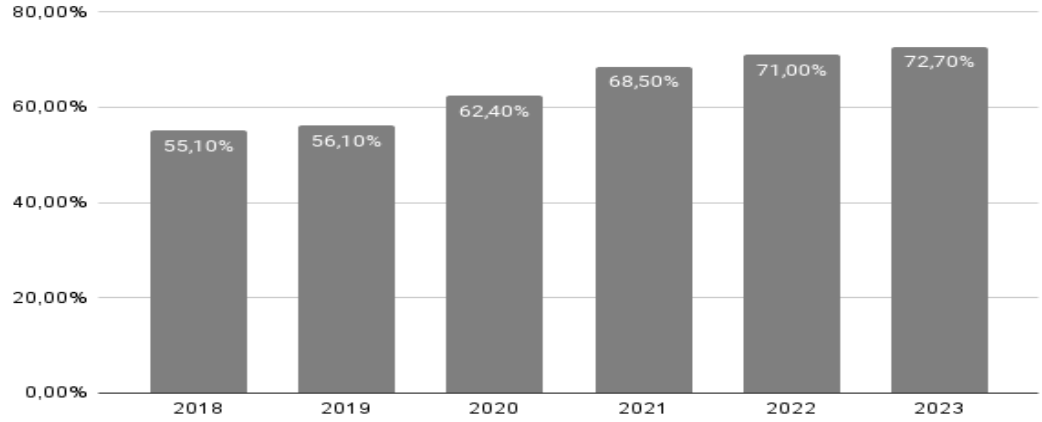

Figure 1: Growth in the number of cyberattacks over the years in million [7].

Polymorphic malware is a type of virus that can change its code while retaining its core functionality. These viruses usually have a mutation mechanism based on code obfuscation, packaging, and metamorphism techniques that can encrypt or decrypt the virus code, each time creating a unique program code [8]. This adaptive behavior makes static signature-based detection methods ineffective because the malware code differs with each iteration of infection. Thus, the need for dynamic and proactive detection and remediation methods to combat polymorphic malware has become more important than ever. Polymorphic viruses use several adaptive strategies to ensure that they are not detected and neutralized. One of the most common strategies is code encryption using unique encryption algorithms [9]. This encryption

makes it difficult for antivirus software to detect the virus because it looks like a harmless file. A well-known block diagram of polymorphic malware detection is shown in Figure 3.

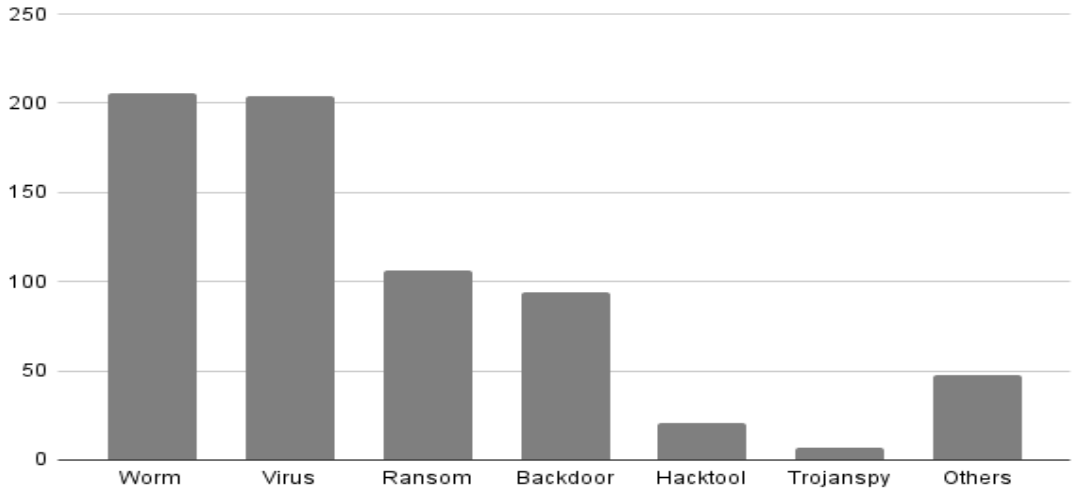

Figure 2: Statistics of the number of cyberattacks by types of malicious software [7].

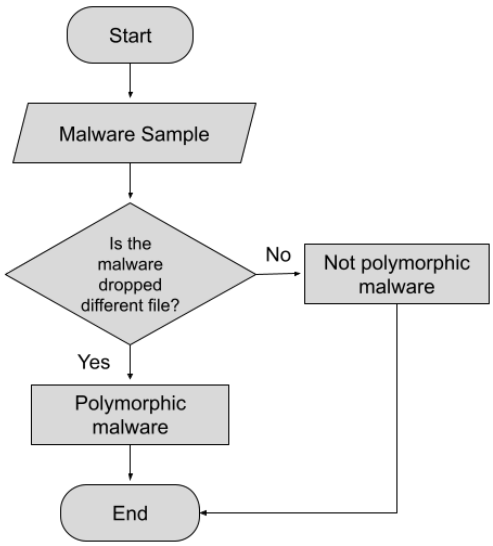

Figure 3: Block diagram of polymorphic malware detection.

In addition, the virus may use an unzip program that only runs when the file is opened, making it more difficult to detect. Finally, polymorphic malware often uses anti-analysis techniques to thwart reverse engineering attempts. This may include methods such as code obfuscation, procedures to prevent reverse engineering, and others [10]. By applying these techniques, polymorphic malware becomes even more elusive, making detection and analysis quite a challenge.

Detection of polymorphic malware requires the use of a combination of static and dynamic analysis methods [11]. While static analysis can provide initial insight into malware behavior, it is often ineffective due to the rapid change of polymorphic malware code. Therefore, dynamic analysis methods are important for effective threat detection and neutralization. Dynamic analysis involves running malware in a controlled environment, such as a virtual machine or sandbox, to observe its behavior [12]. By monitoring system actions, file modifications, network connections, and other indicators, security analysts can identify suspicious behavior and classify malware accordingly [3]. Behavioral analysis techniques are often used to improve detection capabilities. These methods include monitoring the file's runtime behavior, analyzing its actions, and assessing the risks it poses. By comparing the behavior of a potentially malicious executable against known patterns and heuristics, security tools can quickly identify instances of polymorphic malware. In addition, machine learning algorithms play an important role in detecting polymorphic malware. By learning from models and large datasets of known malware, these algorithms learn to identify malicious files and distinguish between polymorphic malware and legitimate software. This approach provides an efficient and scalable solution to combat the ever-growing threat of polymorphic malware. As polymorphic malware continues to evolve and evade traditional methods of detection and remediation, implementing effective countermeasures becomes an increasingly urgent need. Failure to mitigate the threat of polymorphic malware can lead to catastrophic consequences such as data leakage, financial loss, and reputational damage.

2.1. String searching algorithms

The malware detection method is an effective method used in cyber security to detect potential malware in a system [13, 14]. It involves scanning binary code or application code to look for specific lines of data commonly associated with malware.

One of the most common tools for finding strings is the strings command on Unix-based systems. This command scans the file and outputs any sequences of printed characters, which often indicate human-readable lines of code in a program.

In the context of malware detection, these lines can provide valuable information about the potential behavior of a suspicious file. For example, they can detect suspicious API calls, file paths, URLs, or registry keys that are often associated with malicious activity.

However, string searching is not a reliable method. Advanced malware writers often use obfuscation techniques to hide their strings, or they may avoid using suspicious strings altogether. In addition, legitimate programs may also contain suspicious-looking strings by accident.

Therefore, while string searches can be a useful first step in malware analysis, it is important to confirm the results using other methods. This can include dynamic analysis (observing the program's behavior at runtime), static analysis (examining the program's code without running it), or heuristic analysis (comparing the program's behavior or code patterns with known malware signatures).

As such, the method of searching for malware strings is a valuable tool in the cybersecurity analyst's arsenal, but it should be used as part of a broader, comprehensive approach to malware detection and analysis.

2.2. Intelligent data analysis

One of the most promising ways to detect malware is the use of data analysis methods. These techniques involve analyzing large data sets to identify patterns, associations, or anomalies that may indicate malicious activity [15, 16].

The first step in the data mining discovery method is data collection. This involves collecting a wide range of data, such as network traffic logs, tracking system calls and user actions. Data can be collected from a single machine or a network of computers for broader analysis.

Once data is collected, it is often pre-processed to convert it into a suitable format for data analysis. For example, raw data may need to be converted to a numeric format or filtered out for irrelevant data.

Then, the pre-processed data is subjected to data mining algorithms. There are several types of data mining techniques that can be used, including classification, clustering, regression, and anomaly detection. These techniques can help identify patterns or anomalies that may indicate the presence of malware.

Finally, the results can be presented in a format that is easily interpreted by computer security analysts, such as a visual dashboard or notification system.

Classification, for example, involves training a model to recognize the characteristics of known malware and then using that model to classify new data as safe or malicious. Clustering, on the other hand, groups similar data together, which can help identify patterns in the data that may indicate an attack.

After the data mining process, the results are often post-processed to remove any false positives or negatives. This may include cross-checking the results with other detection methods or manually checking for malware detection.

It's worth noting that while data mining can be a powerful tool for malware detection, it's not foolproof. Sometimes it can give false positives or give a negative response. Therefore, it cannot detect all types of malware. However, when combined with other detection methods, data mining can significantly improve a system's ability to detect and respond to malware threats.

2.3. Sandbox analysis

Malware sandbox analysis is a technique used by cybersecurity professionals to analyze and understand the behavior of malware in a controlled environment [17, 18]. It involves running malware in a virtual or isolated environment, known as a sandbox, to observe its activities and gather valuable information.

The goal of malware analysis is to reveal the capabilities of the malware, identify potential threats, and develop effective countermeasures. By executing malware in a controlled environment, analysts can study its interactions with the operating system, network, and other software components.

During the analysis, various dynamic and static techniques are used. Dynamic analysis includes monitoring the malware's runtime behavior, such as file system modifications, network communication, and system calls. Static analysis, on the other hand, focuses on examining the code and structure of the malware without execution.

Information gathered from analyzing the behavior of a malicious program in an isolated software environment helps identify infection vectors, infrastructure and management practices, payload delivery mechanisms, and potential data theft methods. This knowledge is critical to developing effective detection methods, updating security controls, and mitigating the impact of malware attacks.

In summary, analysis in an isolated software environment is an important component of modern cybersecurity practices. It provides valuable information about the behavior and characteristics of malware, allowing cybersecurity organizations to improve their defense mechanisms and develop forward-looking methods to counter new threats.

Traditional malware detection methods often struggle to keep up with the rapidly evolving malware attack landscape. Machine learning techniques have become a powerful tool to improve malware detection and combat these threats.

2.4. Machine learning algorithms

Machine learning algorithms can analyze large amounts of data and extract patterns and features that can be used to detect malicious behavior [19, 20, 21]. By training models on known malware samples and legitimate software, machine learning algorithms can learn to distinguish between them and accurately classify new and unknown files.

One of the key benefits of using machine learning to detect malware is its ability to adapt and learn from new threats. As new types of malware emerge, machine learning models can be updated and retrained to effectively detect these new threats.

There are several approaches to malware detection using machine learning, including static analysis and dynamic analysis. Static analysis involves examining the code and structure of a file without executing it, while dynamic analysis involves running the file in a controlled environment to observe its behavior. Both approaches can provide valuable information for malware detection.

Cesare and Xiang proposed a polymorphic malware classification method called Malwise (Figure 4), which uses program-level emulation to unpack the malware code [22].

However, it is important to note that detecting malware using machine learning is not without challenges. Adversarial attacks, where attackers manipulate malware to avoid detection, can pose a significant problem. In addition, the large volume of data and the need to constantly update and retrain models require significant computing resources.

In summary, machine learning offers promising solutions for malware detection by leveraging its ability to analyze vast amounts of data and identify patterns. By constantly improving and updating models, machine learning can improve the security of computer systems and networks against new malware threats.

2.5. Structural feature engineering

Structural feature engineering is a key aspect of the development of effective malware detection models [23-25].

By extracting meaningful features from structured data, data analysts and researchers can improve the accuracy and reliability of their malware detection systems.

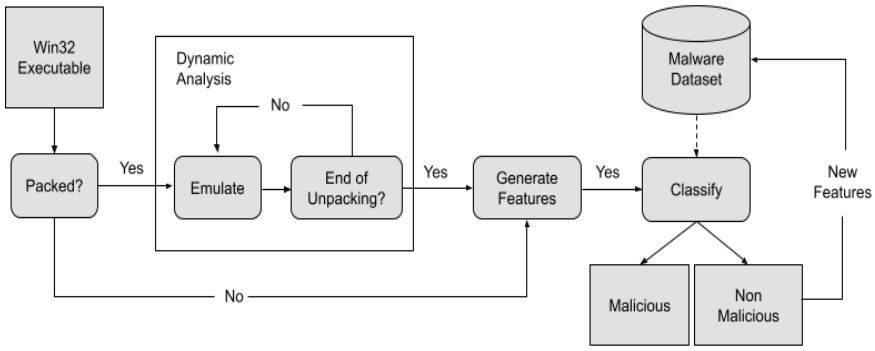

Figure 4: Block diagram of the malware classification system [22].

The following steps describe a structural feature development method specifically designed for malware detection:

1. Understanding data: Gaining a complete understanding of the structure and characteristics of malware data. Identifying relevant variables, their types, and any patterns or relationships present in the dataset.
2. Feature Identification: Identifying features that may be informative for malware detection. This can be achieved through domain knowledge, exploratory data analysis, or statistical techniques specifically designed for malware detection.
3. Feature Extraction: Extracting selected features from raw malware data and converting them into a suitable format for analysis. Application of mathematical transformations, scaling, normalization or encoding methods for preprocessing functions.
4. Feature building: Creating new features by combining or modifying existing features in a way that captures important aspects of malware behavior. This may include aggregations, mathematical operations, or interactions between variables.
5. Feature Selection: Selecting the most relevant features that significantly contribute to malware detection. This helps to reduce the dimensionality and improve the efficiency and accuracy of the detection model.
6. Coding of features: coding of categorical features into numerical representations that can be processed by machine learning algorithms. Use techniques such as single coding, label coding, or target coding to effectively represent categorical variables.
7. Scaling functions: Scale functions to a common range to ensure that they have comparable magnitudes. Standardization, normalization methods can be used for this.
8. Feature Validation: Validate the developed features by evaluating their performance in a malware detection model. Using methods such as cross-validation and model evaluation metrics to measure the performance of the developed features and iteratively improve them as needed.

By following this method of developing structural features, analysts and data scientists can improve the accuracy and reliability of their malware detection systems, leading to improved cybersecurity and anti-malware measures.

The disadvantages of the considered methods require new approaches to the detection and analysis of malicious software. Among them is the detection of malware using probabilistic logic networks (PLN).

3. Methodology

3.1. Probabilistic logic networks (PLN)

Malware detection is a critical aspect of cyber security. PLN [26-28] offer a powerful approach to detect and mitigate malware threats. PLNs combine probabilistic reasoning with logical inference to model complex relationships and dependencies in malware detection.

PLN is a hybrid framework that combines probabilistic graphical models with first-order logic. They provide a flexible and expressive representation for capturing uncertainty and reasoning about complex domains. PLNs utilize the strengths of both probabilistic reasoning and logical inference, making them suitable for malware detection.

One of the key advantages of PLNs in malware detection is their ability to handle uncertain and incomplete information. By assigning probabilities to different hypotheses, PLNs can estimate the probability of the presence of malware and make informed decisions. This probabilistic reasoning allows for more accurate and adaptive detection mechanisms.

PLNs excel at capturing complex malware behaviors and patterns. They can represent both static and dynamic characteristics of malware, including code structure, system interactions, and propagation mechanisms. By modeling this behavior, PLNs can effectively distinguish between legitimate and malicious software.

To train PLN to detect malware, a large dataset of known malware samples and benign software is required. Machine learning methods can be used to study PLN parameters and structure from these data. By iteratively refining the PLN with training examples, it can be tuned to accurately detect and classify PWDs.

Advantages of PLN for malware detection:

- flexibility: PLNs provide a flexible framework for modeling and justifying malware behavior, allowing for adaptation to new threats;
- processing uncertainty: the probabilistic nature of PLN allows processing uncertain and incomplete information, increasing the accuracy of malware detection;
- expressiveness: PLNs can capture complex relationships and dependencies found in malware, providing more comprehensive detection capabilities;
- training from data: PLN can be trained using machine learning techniques, allowing for continuous improvement based on new malware samples.

Challenges in PLN for malware detection:

- scalability: as the complexity of malware and the size of datasets increase, scaling PLN to handle large-scale detection becomes a challenge;
- knowledge development: creating a knowledge base and defining logical rules for detecting malicious software requires experience and knowledge in the field;
- computational complexity: performing inference and learning in PLN can be computationally demanding, requiring efficient algorithms and systems.

3.2. A comprehensive approach to the detection and analysis of polymorphic malware

In the study for the detection of polymorphic malicious software, a complex approach (Figure 5) is proposed, which consists of 3 stages. The first one uses string search algorithms. The second is a complex of methods, including intelligent data analysis, sandbox analysis, machine learning, and structural feature engineering. In the third step, the use of PLN is proposed, which will allow establishing the probability of the software belonging to polymorphic malware. The use of the proposed integrated approach will also allow to determine the necessary methods for neutralization of detected malicious software.

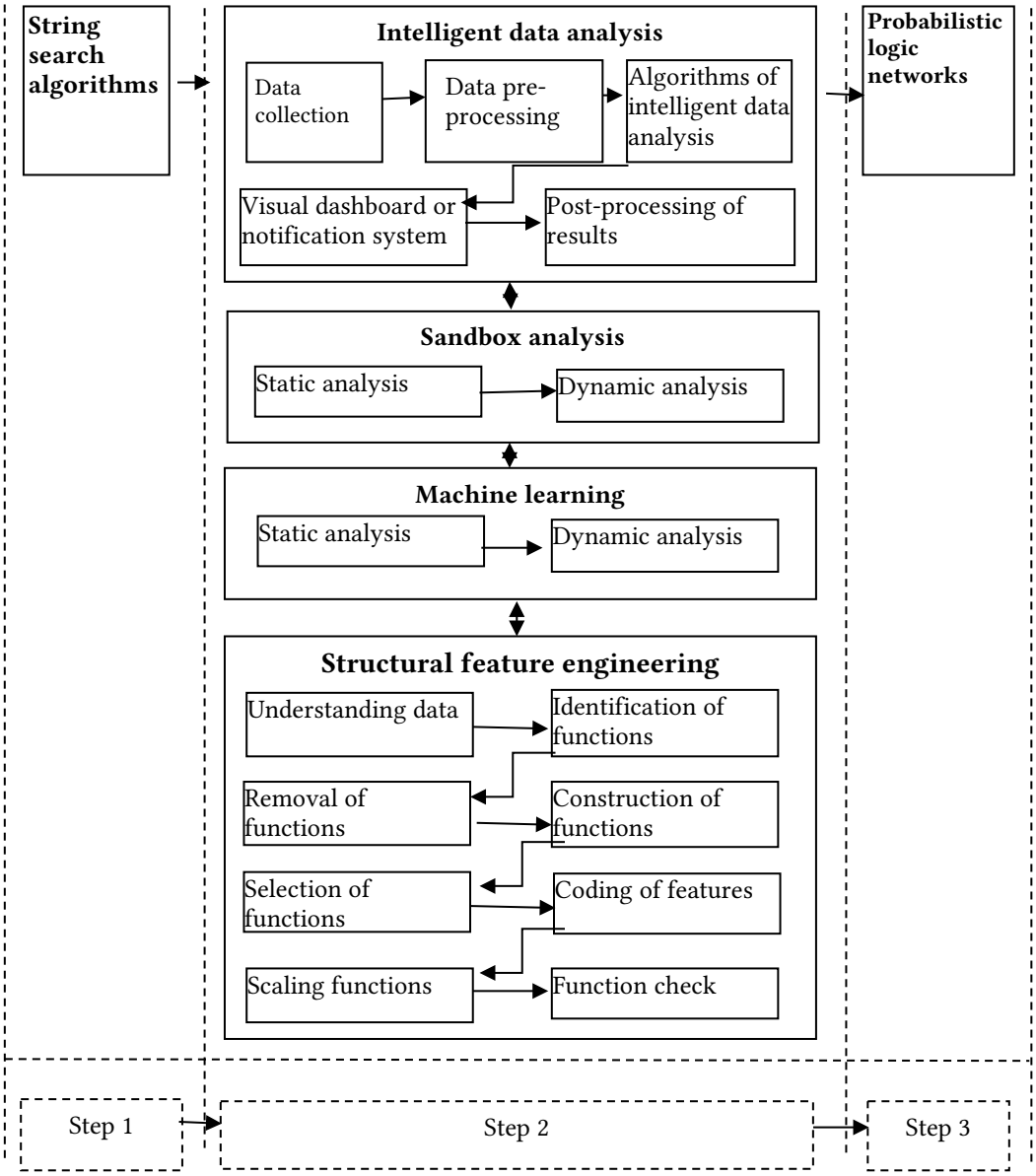

Figure 5: A comprehensive approach to the detection and analysis of polymorphic malware.

4. Experiments

A series of experiments was conducted to determine the effectiveness of the proposed technique. Various types of polymorphic generators were used to obtain modified polymorphic versions of viruses taken from [29]. All polymorphic versions, the generators they created were compiled with anti-debugging and anti-emulation options. For the first experiment, 100 viruses were generated. To evaluate the effectiveness of the proposed method, the percentage of detected viruses was determined at each step of the comprehensive approach proposed in the study.

The results of the conducted experiment are shown in Table 1.

Thus, only 12% of viruses were detected in step 1, 61% in step 2, and 89% in step 3 using PLN. The effectiveness of the proposed method according to the conducted experiment is 28% due to the use of PLN. Also, of the 89% of viruses detected by PLN, 9% were assigned to the range of probability of belonging to malicious software at the level of 0-25% (low level), at the level of 25-75% (medium level) - 19%, at the level of 75- 100% - 72% (high level). The use of PLN allowed not only to increase the effectiveness of malware detection, but also to classify by the level of probability of belonging to malicious software.

5. Conclusions

The study proposes a comprehensive approach to the detection and analysis of polymorphic malware. This approach consists of three stages. The first one uses string search algorithms. The second is a complex of methods, including intelligent data analysis, sandbox analysis, machine learning and structural feature engineering. In the third step, the use of PLN is proposed, which will allow establishing the probability of the software belonging to polymorphic malware. The effectiveness of the proposed method according to the conducted experiment is 28% due to the use of PLN. The use of PLN allowed not only to increase the effectiveness of malware detection, but also to classify by the level of probability of belonging to malicious software.

Table 1

The percentage of detected viruses at each step of the proposed integrated approach

Number of viruses generated	The percentage of viruses detected by string search algorithms (step 1)	The percentage of detected viruses by the methods of step 2	Percentage of viruses detected using PLN (step 3)	The range of probability that viruses belong to polymorphic malware	The number of viruses in the range of probability of belonging to malware
100	12 %	61 %	89 %	0-25 % (low)	9 %
				25-75 % (medium)	19 %
				75-100 % (high)	72 %

References

[1] A. Kashtalian, S. Lysenko, O. Savenko, A. Nicheporuk, T. Sochor, V. Avsiyevych, Multi-computer malware detection systems with metamorphic functionality, Radioelectronic and Computer Systems 1 (2024) 152-175. doi: 10.32620/reks.2024.1.13.

[2] G. Markowsky, O. Savenko, S. Lysenko, A. Nicheporuk, The technique for metamorphic viruses' detection based on its obfuscation features analysis, CEUR-WS 2104 (2018): 680–687.

[3] O. Pomorova, O. Savenko, S. Lysenko, A. Nicheporuk, Metamorphic Viruses Detection Technique based on the the Modified Emulators, CEUR-WS 1614 (2016) 375-383.

[4] O. Savenko, S. Lysenko, A. Nicheporuk, B. Savenko, Approach for the Unknown Metamorphic Virus Detection, in: Proceedings of the 8-th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS, Bucharest, Romania, 2017, pp. 71–76. doi: 10.1109/IDAACS.2017.8095052

[5] B. Savenko, A. Kashtalian, A method for determining the effectiveness of a distributed system for detecting abnormal manifestations, Computer Systems and Information Technologies 2 (2022) 14–22. doi: 10.31891/csit-2022-2-2 In Ukrainian

[6] A. Damodaran, F.D. Troia, C.A. Visaggio, T. H. Austin, M. Stamp, A comparison of static, dynamic, and hybrid analysis for malware detection, J Comput Virol Hack Tech 13 (2017) 1–12. doi: 10.1007/s11416-015-0261-z

[7] Statistic Data. URL: https://www.statista.com/.

[8] K. Brezinski, K. Ferens, Metamorphic Malware and Obfuscation: A Survey of Techniques, Variants, and Generation Kits, Security and Communication Networks, 2023 (2023) 8227751. doi: 10.1155/2023/8227751.

[9] M. Singh, A. Carlson, Exploring Polymorphic Algorithms and Their Use in Cryptography, in: Proceedings of the 2024 IEEE 14th Annual Computing and Communication Workshop and Conference, CCWC, Las Vegas, NV, USA, 2024. doi: 10.1109/CCWC60891.2024.10427812.

[10] B. Anderson, D. McGrew, OS fingerprinting: New techniques and a study of information gain and obfuscation, in: Proceedings of the 2017 IEEE Conference on Communications and Network Security, CNS, Las Vegas, 2017, pp. 1–9. doi: 10.1109/CNS.2017.8228647

[11] L. Bilge, Y. Han, M. Dell'Amico, Riskteller: Predicting the risk of cyber incidents, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, Dallas, Texas, USA, 2017, pp. 1299-1311. doi: 10.1145/3133956.3134022

[12] U. Urooj, B.A.S. Al-rimy, A. Zainal, F.A. Ghaleb, M.A. Rassam, Ransomware Detection Using the Dynamic Analysis and Machine Learning: A Survey and Research Directions. Applied Sciences, 12, (2022) 172. doi: 10.3390/ app12010172

[13] K. Gundogan, K. Gupta, L. Garland, C. Varol, N. Shashidhar, Identifying Malware Family with String Matching Algorithms Based on API Calls and Entire Strings, in: Proceedings of the 12th International Symposium on Digital Forensics and Security ,ISDFS, San Antonio, TX, USA, 2024. doi: 10.1109/ISDFS60797.2024.10527225.

[14] Zh. Zhang, Review on String-Matching Algorithm, SHS Web of Conferences 144 (2022) 03018. doi: 10.1051/shsconf/202214403018

[15] H. Sayadi, Z. He, H. M. Makrani, H. Homayoun, Intelligent Malware Detection based on HardwarePerformance Counters: A Comprehensive Surve, in: Proceedings of the 25-th International Symposium on Quality Electronic Design, ISQED'24, San Francisco, California, 2024. doi: 10.1109/ISQED60706.2024.10528369

[16] R. Beg, R.K Pateriya, D. S. Tomar, ACMFNN: A Novel design of an augmented convolutional model for intelligent cross-domain malware localization via forensic neural networks, IEEE Access XX (2017). doi: 10.1109/ACCESS.2023.3305274

[17] Z. Balazs, Malware Analysis Sandbox TestingMethodology The Journal on Cybercrime & Digital Investigations 1, 1 (2015). doi: 10.18464/cybin.v1i1.3

[18] B. Sun, A. Fujino, T. Mori, T. Ban, T. Takahashi, D. Inoue, Automatically Generating Malware Analysis Reports UsingSandbox Logs, IEICE transactions on information and systems E101–D, 11 (2018) 2622-2632. doi: 10.1587/transinf.2017ICP0011

[19] R. Chiwariro, L. Pullagura, Malware Detection and Classification Using Machine Learning Algorithms, International Journal for Research in Applied Science & Engineering Technology, IJRASET, 11 (2023) 1727-1738. doi: 10.22214/ijraset.2023.55255

[20] A. J. Kurian, A. Santhosh, M. Subin, Enhanced malware detection framework leveraging machine learning algorithms, International Research Journal of Modernization in Engineering Technology and Science 06(03) (2024) 3597-3603.

[21] I. Obeidat, M. AlZubi, Developing a faster pattern matching algorithms for intrusion detection system. *International Journal of Computing*, *18*(3), 2019, 278-284. doi:10.47839/ijc.18.3.1520

[22] S. Cesare, Y. Xiang, Classification of malware using structured control flow, in: Proceedings of the 8-th Australasian Symposium on Parallel and Distributed Computing, AusPDC 2010, Brisbane, Australia, 107, 2010, pp. 61-70. doi: 10.5555/1862294.1862301

[23] E. Masabo, K.S. Kaawaase, J. Sansa-Otim, D. Hanyurwimfura, Structural Feature Engineering approach for detecting polymorphic malware, in: Proceedings of the 15-th IEEE Intl Conf on Dependable, Autonomic and Secure Computing, 15-th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress, DASC/PiCom/DataCom/CyberSciTech, 2017, pp. 716-721. doi: 10.1109/DASC-PICom-DataCom-CyberSciTec.2017.125

[24] Y. T. Ling, · N. F. M. Sani, · M. T. Abdullah, · N. A. W. A. Hamid, Metamorphic malware detection using structural features andnonnegative matrix factorization with hidden markov model, Journal of Computer Virology and Hacking Techniques 18 (2022)183–203. doi: 10.1007/s11416-021-00404-z

[25] Y. T. Ling, N. F. M. Sani, M. T. Abdullah, N. A. W. A. Hamid, Structural Features with Nonnegative Matrix Factorization for Metamorphic Malware Detection, Computers & Security 104, 2 (2021) 102216. doi: 10.1016/j.cose.2021.102216

[26] M. Qu, J. Tang, Probabilistic Logic Neural Networks for Reasoning, in: Proceedings of the 33-rd Conference on Neural Information Processing Systems, NeurIPS 2019, Vancouver, Canada, 2019. doi: 10.48550/arXiv.1906.08495

[27] K. M. M. Sadeghi, B. Goertzel, Uncertain Interval Algebra via fuzzy/probabilistic modeling, in: Proceedings of the 2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Beijing, China, 2014. doi: 10.1109/FUZZ-IEEE.2014.6891863

[28] C. Harrigan, B. Goertzel, M. Ikle, A. Belayneh, G. Yu, Guiding Probabilistic Logical Inference with Nonlinear Dynamical Attention Allocation, Lecture Notes in Computer Science 8598 (2014) 238-241.

[29] VX Heavens [online] URL: http://vxheaven.org/