Detecting software implants using system decoys*

Dmytro Denysiuk[1,*,†], Oleg Savenko [1,†] , Sergii Lysenko[1,†], Bohdan Savenko[1,†] and Andrii Nicheporuk[1,†]

[1] *Khmelnytskyi National University, Institutska str., 11, Khmelnytskyi, 29016, Ukraine*

Abstract

This paper presents a new method for detecting software implants based on the use of software decoys and in-depth analysis of system parameters. The aim of the study was to compare the effectiveness of the proposed method with existing approaches, such as signature analysis, behavioral analysis, and machine learning-based methods. For this purpose, a relevant dataset was collected, including 5000 malware samples and 5000 legitimate programs. Each sample was analyzed for detailed signs of interaction with the file system, RAM, process behavior, and network activity. The research methodology included data collection and labeling, feature extraction and normalization, and the use of recurrent neural networks (RNNs) to analyze complex behavioral patterns. The proposed method used software decoys to attract malware, which allowed detecting its activity at early stages. Experiments showed that the method achieves 95% accuracy, 94% completeness, 96% prediction accuracy, and 95% F1-measure, which significantly exceeds the performance of signature analysis (85% accuracy), behavioral analysis (89% accuracy), and machine learning methods (91% accuracy). The proposed approach has several key advantages: the active use of software decoys increases the likelihood of detecting threats, in-depth analysis of system parameters provides a comprehensive overview of program behavior, and the use of RNNs allows recognizing complex and unknown patterns. In addition, the method demonstrates a high detection rate, which makes it suitable for use in real-time systems. The results of the study indicate the high potential of the proposed method for improving the cybersecurity of modern information systems. The method can be integrated into existing protection systems, such as intrusion detection systems (IDS) and SIEM systems, providing a more efficient and prompt response to cyber threats. In future research, it is planned to expand the dataset and optimize the model to reduce computational costs, as well as conduct testing in real-world environments to assess the practical effectiveness of the method. Thus, the proposed method represents a significant step forward in the field of software implant detection, providing high accuracy, completeness and speed of detection, which is critical for protecting information systems from modern and evolving cyber threats.

Keywords

malware detection, software implants, software decoys, behavioral analysis, machine learning, deep learning, recurrent neural networks, cybersecurity, intrusion detection systems (IDS).

1. Introduction

The proliferation of software implants [1], such as malware, rootkits[2], and backdoors[3], poses a significant threat to the information security of modern computer systems. These malicious components hidden in software are capable of unauthorized access to system resources, stealing confidential information, and compromising data integrity. The increasing complexity and sophistication of software implants makes them difficult to detect using traditional methods based on signature analysis or simple anomaly detection.

A software implant is a malicious code [4] or module that is secretly installed on computer systems or devices to gain unauthorized access, collect confidential information, or perform other destructive actions without the user's knowledge. They are often used as part of sophisticated

AdvAIT-2024: 1st International Workshop on Advanced Applied Information Technologies, December 5, 2024, Khmelnytskyi, Ukraine - Zilina, Slovakia

* Corresponding author.

† These authors contributed equally.

denysiuk@khmnu.edu.ua (D. Denysiuk); savenko_oleg_st@ukr.net (O. Savenko); sirogyk@ukr.net (S. Lysenko); savenko_bohdan@ukr.net (B. Savenko); nicheporuka@khmnu.edu.ua (A. Nicheporuk)

0000-0002-7345-8341 (D. Denysiuk); 0000-0002-4104-745X (O. Savenko); 0000-0001-7243-8747 (S. Lysenko); 0000-0001-5647-9979 (B. Savenko); 0000-0002-7230-9475 (A. Nicheporuk)

cyberattacks, such as advanced persistent threats (APTs) [5], providing long-term covert access to compromised systems.

Software implants can be introduced into a system through various methods, including exploitation of vulnerabilities [6] in software, social engineering, or infected updates. They are able to operate undetected for a long time, carrying out malicious operations without detection, which is a particular danger for organizations, as it can lead to significant financial losses, loss of reputation, and leakage of confidential information.

One of the key challenges in detecting software implants is their ability to bypass security controls. Modern implants can use stealth techniques, such as rootkits, which modify the operating system kernel or inject code into legitimate processes. This allows them to evade detection by antivirus programs and other security tools that rely on checking known signatures or detecting abnormal behavior.

Additionally, the development of obfuscation [7] and polymorphism [8] techniques allows malware to change its code or behavior, making it difficult to detect [9] even with advanced analyzers. This underscores the need to develop new methods that are independent of prior knowledge of malware and can effectively respond to new threats.

One of the most promising areas in the fight against software implants is the use of system decoys[10] that act as traps for malware. System decoys can be implemented in the form of specially created files, processes, or network services that imitate vulnerable or attractive objects for attackers. When malware interacts with such decoys, its presence is detected and the threat is neutralized.

Compared to traditional methods, the use of system decoys has several advantages. First, these tools do not rely on known signatures[11] or malware behavioral patterns, making them effective against new or modified threats. Secondly, decoys can be integrated at different levels of a system, providing multi-level protection. Thirdly, interaction with the decoy can help collect additional information about the malware, allowing for more detailed analysis and development of countermeasures.

2. Overview of existing solutions

Honeypot systems continue to play an important role in ensuring the cybersecurity of modern computer networks. They function as specialized tools that simulate[13] real systems or services in order to attract attackers, allowing cybersecurity professionals to investigate their methods and techniques in detail. Between 2021 and 2024, there have been significant advances in the development of honeypot architectures and related software, including integration with artificial intelligence[13] and machine learning[14] technologies to improve threat detection and analysis. These systems not only help identify potential threats, but also provide in-depth analysis of attackers' actions in various environments, including cloud computing, the Internet of Things (IoT), industrial cyber-physical systems (CPS)[15], and traditional network infrastructures. In addition, the improvement of honeypot systems contributes to the formation of more adaptive and proactive cyber defense strategies, which is important in the context of the growing complexity and scale of modern cyber threats.

2.1. Main categories of honeypot systems

Honeypot systems are divided into two main types depending on the level of interaction: low interaction and high interaction. Low-interaction systems, such as Honey, are limited to emulating a limited set of services. They involve only basic attack attempts such as port scans or entry-level exploits. Although these systems are less resource-intensive, they are not capable of investigating sophisticated attack methods in detail. In contrast, highly interoperable systems, such as Dionaea or Kippo, offer attackers full operating systems or real services to interact with. This allows attackers to perform more complex operations while remaining isolated from critical systems. Thanks to this, experts can get more information about the penetration methods used by cybercriminals. T-Pot, for example, is a comprehensive platform that integrates several honeypot solutions and provides in-

depth real-time analysis. The system has been active in recent years and has received updates aimed at improving performance and monitoring capabilities.

2.2. Honeynet systems

Honeynet systems[16] consist of several honeypot services, which allows you to simulate an entire network infrastructure, including servers, databases, and other important elements of corporate systems. One important example of a modern honeynet architecture is HoneyFactory, which uses container technologies to create virtual environments. This solution provides fast deployment of complex network systems and enhances attack detection capabilities through the use of cyber detection. Compared to previous versions of honeynet, HoneyFactory shows better results in terms of protection efficiency due to the high speed of request processing and flexible system settings for different business needs.

The use of honeynet systems has become popular in various environments, including IoT and CPS. These technologies allow you to protect not only traditional network environments, but also new-generation infrastructures, where it is important to monitor both internal and external threats. Recent studies have emphasized the importance of integrating such systems into critical infrastructure to obtain enhanced information about attack methods and their prevention.

2.3. Honeytoken, Honeypatch та Honeyclient

In addition to honeypot systems, other decoys are being actively developed that perform additional functions in threat detection. Honeytoken[17] is one of the most common tools for detecting unauthorized activities on the network. Programs such as Canarytokens allow you to create decoy files that automatically generate alerts when they are accessed. For example, a file that looks like an important document can be a signal to detect cybercriminals trying to read or modify it.

Honeypatch, introduced in 2023, is an innovative technology that allows you to test the security of systems without risking productive environments. It creates vulnerable components that attackers can attack, allowing you to study their behavior and find new threats. This method is effectively used to collect information about attacks and to test the readiness of systems to exploit vulnerabilities.

Honeyclient systems, such as Capture-HPC, are used to detect threats targeting client applications. They actively interact with potentially malicious websites and analyze the methods used to infect client applications. This technology allows you to effectively simulate real user behavior and detect attacks such as drive-by downloads.

Thus, the development of malware detection systems using decoys is a promising and highly sought-after area. The use of such technologies allows not only to detect and analyze modern cyber threats more effectively, but also to predict possible attacks, increasing the overall level of security of information systems. Further development and implementation of decoy software will help create more adaptive and proactive protection strategies, which is important in the context of the ever-increasing complexity and dynamics of malware.

3. Detection of software implants

Successful development of a malware decoy detection model requires an in-depth analysis of the parameters that the system will monitor. Identifying these parameters is key to effectively detecting and analyzing malicious activity on the system. The main aspects that need to be considered in detail include file system interaction, changes in RAM usage, process behavior, and network activity. Understanding the behavioral patterns typical of software implants is critical to developing an effective model.

Successful development of a malware decoy detection model requires an in-depth analysis of the parameters that the system will monitor. Identifying these parameters is key to effectively detecting and analyzing malicious activity on the system. The main aspects that need to be considered in detail

include file system interaction, changes in RAM usage, process behavior, and network activity. Understanding the behavioral patterns typical of software implants is critical to developing an effective model.

Analyzing such patterns includes tracking the frequency and types of file operations, monitoring changes in directory structure, and detecting unusual or suspicious changes in file sizes. To quantify anomalies in the file system, you can use the anomaly indicator A_f:

$$A_f = \sum_{i=1}^{N} \omega_i \cdot \left(\frac{f_i - \mu_i}{\sigma_i}\right), \quad (1)$$

where:

- f_i – frequency of the operation i,;
- μ_i and σ_i – average value and standard deviation of the frequencyf_i in the normal state of the system;
- ω_i – weighting factor for the operation i;

It is especially important to pay attention to operations with system files, configuration files, and the registry, as changes to them may indicate attempts to compromise the system.

Changes in the operation of RAM are another significant indicator of a potential threat. Software implants can load their code directly into memory, bypassing the file system, or inject it into the memory of other processes, making them difficult to detect using traditional methods. Analysis of memory usage patterns includes monitoring the creation of new memory segments, changes in access rights to them, and analyzing the contents of memory for malicious signatures or abnormal data structures.

To quantify changes in memory usage, you can consider the rate of change in the amount of memory used:

$$\Delta M = \frac{dM_t}{dt} \quad (2)$$

where M_t – the amount of memory used at a given time$t$. If the value ΔM exceeds the threshold valueT_M, this may indicate abnormal activity. For example, the detection of executable code in memory areas that usually do not contain such code can be described through the indicator function I_{det}:

$$I_{det}(x) = \begin{cases} 1, & if\ memory\ location\ x\ contains\ executable\ code \\ 0, & otherwise \end{cases}, \quad (3)$$

If $I_{det}(x) = 1$ for the region x, where the executable code is not expected, this may indicate a software implantation.

The behavior of processes in a system also provides important information for detecting malicious activity. Software implants can create new processes, modify existing ones, or interact with them in unusual ways. They may try to gain elevated privileges, change system settings, disable or bypass security features.

Analysis of process behavior patterns includes monitoring the creation and completion of processes, analyzing their interaction, tracking system calls, and resource usage. Logistic regression can be used to estimate the probability that a process is malicious:

$$P(war \mid x) = \frac{1}{1 + e^{-(\beta_0 + \sum_{i=1}^{n} \beta_i x_i)}}, \quad (4)$$

where:

- $x = (x_1, x_2, x_3, \dots, x_i)$ – a vector of features related to the behavior of the process;

- β_i — model coefficients.

For example, a process that unexpectedly makes a large number of system calls related to network activity or file manipulation may have high values for the following attributesx_i, which will increase the likelihood of $P(war \mid x)$. In addition, you should pay attention to processes that run in the background without user interaction or try to hide their presence by changing their attributes.

Network activity of software implants is often one of the most obvious indicators of their presence. They may attempt to establish unauthorized connections to remote servers to transmit collected data, receive commands, or download additional modules. Analysis of network behavior patterns includes tracking the initiation of network connections, analyzing the protocols, ports, and IP addresses used. To quantify anomalies in network activity, you can use the anomaly indicator A_n:

$$A_n = \sum_{j=1}^{M} \omega_j \cdot \left(\frac{n_j - \mu_j}{\sigma_j}\right)^2, \quad (5)$$

where:

- n_j — measure parameter value j (for example, the number of connections to a specific IP address);
- μ_j and σ_j — the average value and standard deviation of this parameter in the normal state;
- ω_j — weighting factor;

For example, suddenly establishing connections to geographically remote or suspicious addresses, using non-standard or high ports, bypassing proxy servers or firewalls can indicate malicious activity. It's also important to analyze the volume and nature of the data being transmitted, including whether confidential information or large amounts of data are being transmitted for no apparent reason.

For in-depth analysis of these patterns, it is necessary to use modern machine learning and artificial intelligence methods. Deep learning algorithms, such as recurrent neural networks (RNNs)[19] or convolutional neural networks (CNNs)[20], can be used to analyze sequences of actions and identify complex dependencies between different system parameters.

For example, a recurrent neural network models a sequence of input data $\{x_1, x_2, x_3, \dots, x_i\}$ by calculating hidden states h_t by the formula:

$$h_t = \varphi\left(W_{xh}{}^{x_t} + W_{hh}{}^{h_{t-1}} + b_h\right), \quad (6)$$

where:

- W_{xh}, W_{hh} — weight matrices;
- b_h — displacement vector;
- φ — activation function ReLU;

Network output y_t can be calculated as:

$$y_t = W_{hy}{}^{h_t} + b_y \quad (7)$$

where:

- W_{hy} — output weight matrix;
- b_y — displacement vector.

Analyzing time series of network activity using RNNs can help detect hidden patterns of communication between malware and command-and-control[21] servers that can be disguised as legitimate traffic. In addition, it is important to consider contextual factors and profiles of normal

system behavior. The use of behavioral analysis allows the model to detect deviations from the norm that may not be obvious when considering individual parameters. The Mahalanobis distance can be used to quantify the deviation[22]:

$$D_M = \sqrt{(x-\mu)^T S^{-1}(x-\mu)} \quad (8)$$

where:

- $x -$ vector of sporasterzhuvannye signs;
- $\mu -$ is a vector of average values of features in the normal state;
- $S -$ is the covariance matrix.

For example, a program that does not perform network activity under normal conditions but suddenly starts sending data to the network[23] may have a significant D_M deviation, indicating an anomaly. Temporal aspects, such as the time of day when certain activities occur or the duration of sessions, should also be considered, which can help identify anomalies.

Software implants often use sophisticated techniques to bypass detection tools, such as polymorphism, metamorphism, code obfuscation, rootkits, and other concealment methods. Therefore, the model must be able to detect not only known signatures or patterns, but also new, previously unknown threats. This can be achieved by using unsupervised learning and clustering methods. One of them, the k-means algorithm[24], allows you to divide data into k clusters by minimizing the sum of squares of the distances between points and cluster centroids:

$$arg\ min_S \sum_{k=1}^{K} \sum_{x_i \in S_k} \|x_i - \mu_k\|^2 \quad (9)$$

where:

- $S_k -$ cluster$k_,$;
- $\mu_k -$ cluster centroid k;

Identifying new behavioral clusters can signal the emergence of new malicious patterns. Integration of the model with existing security and monitoring tools is an important component that provides an expanded picture of the system state and facilitates rapid response to threats. For example, integration with intrusion detection systems (IDS)[25], event log management tools, or SIEM systems[26] provides additional data for analysis, which increases the model's accuracy. Performance and optimization issues are equally important: the model must operate in real time or close to it to ensure timely detection and response to threats[27]. This requires optimization of algorithms and the use of efficient data processing methods, such as streaming processing or hardware acceleration.

4. Results

To evaluate the effectiveness of the proposed method of detecting software implants using software decoys, a detailed experimental analysis was conducted. The purpose of the experiment was to compare the proposed method with existing malware detection methods, such as signature analysis[28], behavioral analysis, and machine learning-based methods[29].

The first step of the experiment was to prepare a relevant dataset that would adequately reflect the real conditions of the system. For this purpose, we collected a large dataset consisting of various types of malware[30] and legitimate programs. Malicious samples included trojans, rootkits, backdoors, spyware, and other types of software implants. These samples were obtained from open sources, such as VirusTotal, MalwareBazaar, and other specialized repositories. To ensure a representative dataset, 5000 samples of malware and 5000 samples of legitimate programs were selected, including system utilities, office applications, browsers, and other legitimate software. Each sample was thoroughly tested for errors and correct operation. An even distribution between the

different types of malware was ensured to avoid bias in the results of the experiment. Next, the data was labeled. Malicious samples were labeled as negative (label "1") and legitimate programs as positive (label "0"). This allowed us to use binary classification methods[31] to analyze the data. For each sample, information was collected on file system interaction, RAM usage, process behavior, and network activity. This data was obtained using specialized monitoring tools such as Sysinternals Suite, Wireshark, and custom software decoys integrated into the system[32]. Special attention was paid to the feature extraction process. About 100 different features were identified for each sample, including:

- File operations - the number of files created, deleted, modified, file types interacted with, changes in attributes and access rights.
- Memory operations - number of memory segments created, changes in memory access rights, amount of memory used, code injections.
- Process behavior - the number of processes created and terminated, the use of system calls, interaction between processes, attempts to gain elevated privileges.
- Network activity - the number of established connections, ports and protocols used, IP addresses, and the amount of data transmitted and received.

To ensure data quality, the features were normalized and scaled. This allowed us to avoid the influence of the scales of various parameters on the modeling results. A correlation analysis was also performed to identify and eliminate redundant data.

After preparing the dataset, a series of experiments was launched to compare the effectiveness of different methods for detecting software implants. The experiments were conducted in a controlled environment using specialized hardware and software.

In the first experiment, we applied signature analysis. For this purpose, antivirus software with up-to-date signature databases was used[33]. The dataset was run through the antivirus and the results were recorded. The signature analysis allowed us to detect most of the known samples of software implants, but showed low efficiency in relation to new or modified samples.

The second experiment involved the use of behavioral analysis. A monitoring system was deployed that analyzed the behavior of programs in real time. This method made it possible to detect malware that exhibited abnormal activity, but had limitations regarding hidden or well-camouflaged software implants. The third experiment was conducted using machine learning methods. The dataset was divided into training and test samples in the ratio of 70/30. Classification algorithms such as logistic regression, SVM, and decision trees were used. The models were trained on the training set and tested on the test set. The results showed better performance compared to previous methods, but still had shortcomings in detecting new types of software implants [34].

In the fourth experiment, the proposed method was applied using software decoys and in-depth analysis of system parameters. Additional software decoys were created to simulate critical system resources[35]. This made it possible to attract software implants and detect their activity at early stages. Deep neural networks were also used to analyze complex behavioral patterns.

The model was trained on the full dataset using cross-validation to improve overall performance. Metrics such as True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) were used to accurately assess the effectiveness of each method. These metrics allow us to analyze the classification results in detail and determine the number of correct and incorrect detections.

Table 1
Classification results for each method

Method	TP	TN	FP	FN
Signature analysis	4000	4500	500	1000
Behavioral analysis	4250	4600	400	750
Machine learning	4400	4650	350	600
Request method	4700	4800	200	300

The TP (True Positive) value shows the number of correctly detected malicious samples. TN (True Negative) indicates the number of legitimate programs correctly identified. FP (False Positive) indicates the number of legitimate programs that were mistakenly[36] recognized as malicious. FN (False Negative) shows the number of malicious samples that were not detected.

Based on these indicators, we calculated the metrics of accuracy, completeness, prediction accuracy, and F1-measure.

Table 2
Comparison of the effectiveness of detection methods

Method	Accuracy (%)	Reproduced (%)	Prediction accuracy (%)	F1-measure (%)
Signature analysis	85	80	88	84
Behavioral analysis	89	85	91	88
Machine learning	91	88	93	90
Request method	95	94	96	95

The results obtained indicate a significant advantage of the proposed method for detecting software implants over traditional approaches. In particular, the proposed method achieved the highest accuracy (95%), completeness (94%), prediction accuracy (96%), and F1-measure (95%). This demonstrates the method's ability to effectively detect both known and new malware samples[37]. The analysis of TP, TN, FP, and FN indicators shows that the proposed method has the lowest number of false positives (FP = 200) and undetected threats (FN = 300) compared to other methods. This is especially important in the context of detecting hidden or well-camouflaged software implants that may go undetected using traditional methods

Comparison with machine learning methods shows that even when using modern algorithms such as logistic regression, SVM, and decision trees, there are limitations in detecting new types of malware. The proposed method, through the use of software decoys and in-depth analysis of behavioral patterns, outperforms these approaches by all major metrics.

Thus, the experimental results confirm the feasibility of implementing the proposed method in cybersecurity systems. It not only improves the detection rate of software implants, but also reduces the risk of missing new or modified threats, which is critical to ensuring the protection of information systems.

5. Discussion

Experimental results show that the proposed method significantly outperforms other methods in all major metrics. In particular, the high number of True Positive (TP) and True Negative (TN) indicates the method's ability to accurately identify both malware and legitimate software. Low values of False Positive (FP) and False Negative (FN) indicate a minimum number of false positives and missed threats, which is critical for cybersecurity.

The analysis of Accuracy shows that the proposed method reaches 95%, which is a significant improvement over signature analysis (85%), behavioral analysis (89%), and machine learning methods (91%). This indicates that an integrated approach that includes the use of software decoys and in-depth analysis of system parameters is more effective in detecting modern complex threats.

The high Recall and Precision values also confirm the effectiveness of the proposed method. The 94% completeness means that the method is able to detect most of the available malicious samples, while the 96% prediction accuracy indicates that most of the detected threats are indeed malicious. This is important to reduce the number of false positives that can divert resources and attention of security professionals.

Detection time is also an important factor. The proposed method provides fast data analysis, which allows detecting threats in almost real time. Compared to the machine learning method, which

requires an average of 1.5 seconds per sample, the proposed method performs analysis in 1.0 seconds, which can be critical in scenarios where response time is critical.

A detailed analysis of the results for different types of malware shows that the proposed method is effective for a wide range of threats. For example, for rootkits, which are usually difficult to detect due to their ability to hide their presence, the method achieved a detection rate of 93%, which is significantly higher than the results of other methods.

The use of software decoys has proven to be particularly effective in detecting software implants that attempt to interact with critical system resources or gain unauthorized access to data. This allows you to detect threats at an early stage, before they can cause significant damage to the system. In addition, the use of deep neural networks to analyze complex behavioral patterns allowed the model to learn to recognize even those threats that use modern detection bypass techniques such as code obfuscation, polymorphism, and metamorphism.

However, it should be noted that the proposed method requires significant computing resources to process a large amount of data and train the model. This can be a challenge for systems with limited resources or in environments where data from a large number of endpoints must be processed.

6. Conclusion

The experimental analysis confirms the high efficiency of the proposed method for detecting software implants using software decoys and in-depth analysis of system parameters. The method demonstrates a significant improvement in all key metrics compared to traditional methods based on signature analysis, behavioral analysis, and machine learning.

The proposed approach allows not only detecting known threats but also effectively detecting new and previously unknown malware that uses sophisticated techniques to bypass detection tools. The use of software decoys provides an additional level of protection, allowing to detect attempts of unauthorized access to critical system resources.

The high accuracy, completeness, and speed of detection make this method promising for use in cybersecurity systems where it is necessary to ensure the maximum level of protection with minimal false positives. In future research, it is advisable to consider optimizing the model to reduce computational costs, as well as conducting real-world testing to assess the practical effectiveness and resistance of the method to various types of attacks.

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly in order to: grammar and spelling check; DeepL Translate in order to: some phrases translation into English. After using these tools/services, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] T. Flügge, J. Kramer, K. Nelson, S. Nahles, and F. Kernen, Digital implantology—a review of virtual planning software for guided implant surgery. Part II: Prosthetic set-up and virtual implant planning, BMC Oral Health, vol. 22, no. 1, p. 23, 2022.

[2] M. Nadim, D. Akopian, and W. Lee, A review on learning-based detection approaches of the kernel-level rootkit, in: 2021 International Conference on Engineering and Emerging Technologies (ICEET), IEEE, 2021, pp. 1–6.

[3] Y. Li, Y. Jiang, Z. Li, and S. T. Xia, Backdoor learning: A survey, EEE Transactions on Neural Networks and Learning Systems, vol. 35(1) (2022) 5–22.

[4] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, A study on malicious software behaviour analysis and detection techniques: Taxonomy, current trends and challenges, Future Generation Computer Systems, vol. 130 (2022) 1–18,.

[5] A. Sharma, B. B. Gupta, A. K. Singh, and V. K. Saraswat, Advanced persistent threats (APT): Evolution, anatomy, attribution and countermeasures, Journal of Ambient Intelligence and Humanized Computing, 14 (7) (2023) 9355–9381.
[6] J. Yin, M. Tang, J. Cao, H. Wang, M. You, and Y. Lin, Vulnerability exploitation time prediction: An integrated framework for dynamic imbalanced learning, World Wide Web, 2022, pp. 1–23.
[7] O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk, A. Nicheporuk. A Technique for detection of bots which are using polymorphic code. Communications in Computer and Information Science. 2014.Vol. 431. PP.265-276, ISSN: 1865-0929.
[8] H. Chakraborty and R. Vemuri, "ROBUST: RTL Obfuscation Using Bi-functional Polymorphic Operators, in: 2024 37th International Conference on VLSI Design and 2024 23rd International Conference on Embedded Systems (VLSID), IEEE, 2024, pp. 499–504.
[9] O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk, K. Bobrovnikova A Technique for the Botnet Detection Based on DNS-Traffic Analysis. Communications in Computer and Information Science. 2015. Vol. 522. PP.127-138, ISSN: 1865-0929.
[10] J. You, S. Lv, Y. Sun, H. Wen, and L. Sun, Honeyvp: A cost-effective hybrid honeypot architecture for industrial control systems, in: ICC 2021-IEEE International Conference on Communications, IEEE, 2021, pp. 1–6.
[11] F. Manders, A. M. Brandsma, J. de Kanter, M. Verheul, R. Oka, M. J. van Roosmalen, et al., MutationalPatterns: The one stop shop for the analysis of mutational processes, BMC Genomics, vol. 23, no. 1, p. 134, 2022.
[12] O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk. Multi-Agent Based Approach for Botnet Detection in a Corporate Area Network Using Fuzzy Logic. Communications in Computer and Information Science. 2013. Vol. 370. PP.243-254, ISSN: 1865-0929.
[13] I. Ahmed, G. Jeon, and F. Piccialli, From artificial intelligence to explainable artificial intelligence in industry 4.0: A survey on what, how, and where, IEEE Transactions on Industrial Informatics, 18 (8) (2022) 5031–5042.
[14] C. Janiesch, P. Zschech, and K. Heinrich, Machine learning and deep learning, Electronic Markets, 31(3) (2021) 685–695,.
[15] D. Moroz, Research of network characteristics of the communication interface of multiprocessor modular systems, Computer Systems and Information Technologies, 3 (2022) 82–90.
[16] K. Zhang, Y. Shi, S. Karnouskos, T. Sauter, H. Fang, and A. W. Colombo, Advancements in industrial cyber-physical systems: An overview and perspectives, IEEE Transactions on Industrial Informatics, 19(1) (2022) 716–729.
[17] L. Tan, K. Yu, F. Ming, X. Cheng, and G. Srivastava, Secure and resilient artificial intelligence of things: A HoneyNet approach for threat detection and situational awareness, IEEE Consumer Electronics Magazine,. 11(3) (2021) 69–78.
[18] V. Papaspirou, L. Maglaras, M. A. Ferrag, I. Kantzavelou, H. Janicke, and C. Douligeris, A novel two-factor honeytoken authentication mechanism, in: 2021 International Conference on Computer Communications and Networks (ICCCN), IEEE, 2021, pp. 1–7.
[19] G. M. Makrakis, C. Kolias, G. Kambourakis, C. Rieger, and J. Benjamin, Industrial and critical infrastructure security: Technical analysis of real-life security incidents, IEEE Access, 9 (2021) 165295–165325.
[20] J. Zhu, Q. Jiang, Y. Shen, C. Qian, F. Xu, and Q. Zhu, Application of recurrent neural network to mechanical fault diagnosis: A review," Journal of Mechanical Science and Technology, vol. 36, no. 2, pp. 527–542, 2022.
[21] N. Ketkar, et al Convolutional neural networks, in: Deep Learning with Python: Learn Best Practices of Deep Learning Models with PyTorch, pp. 197–242, 2021.
[22] M. Chornobuk, V. Dubrovyn, and L. Deineha, Cybersecurity: Research of DDoS detection methods, Computer Systems and Information Technologies, 4 (2023) 6–9.
[23] S. Lysenko, O. Pomorova, O. Savenko, A. Kryshchuk, K. Bobrovnikova DNS-based Anti-evasion Technique for Botnets Detection. Proceedings of the 8-th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Warsaw (Poland), September 24–26, 2015.Warsaw, 2015. Pp. 453–458.

[24] I. Kok, F. Y. Okay, O. Muyanli, and S. Ozdemir, Explainable artificial intelligence (XAI) for Internet of Things: A survey, *arXiv preprint*, arXiv:2206.04800, 2022.
[25] D. Kim, et al, Class scatter ratio based Mahalanobis distance approach for detection of Internet of Things traffic anomalies, Mobile Networks and Applications, (2023) 1–12.
[26] A. Fahim, K and starting means for k-means algorithm, Journal of Computational Science,. 55 (2021) 101445.
[27] P. Maniriho, A. N. Mahmood, and M. J. M. Chowdhury, A systematic literature review on Windows malware detection: Techniques, research issues, and future directions," Journal of Systems and Software, vol. 2023, p. 111921, 2023. doi:10.1016/j.jss.2023.111921.
[28] J. Lansky, et al., Deep learning-based intrusion detection systems: A systematic review, IEEE Access, 9 (2021) 101574–101599.
[29] S. Vladov, Z. Avkurova, V. Lytvyn, and Y. Zhovnir, Analytical neural network system for the helicopter turboshaft engines operating modes classification, International Journal of Computing, 23(3) (2024) 342–359. doi:10.47839/ijc.23.3.3653.
[30] R. Berdibayev, et al., A concept of the architecture and creation for SIEM system in critical infrastructure, in: Systems, Decision and Control in Energy II, Cham: Springer International Publishing, 2021, pp. 221–242.
[31] O. Savenko, S. Lysenko, A. Kryshchuk, Y. Klots. Proceedings of the 7-th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Berlin (Germany), September 12–14, 2013. Berlin, 2013. Pp. 363–368.
[32] Z. Yu, Precision Marketing Optimization Model of e-Commerce Platform Based on Collaborative Filtering Algorithm, Wireless Communications and Mobile Computing 2022 (2022) 1–10.
[33] S. K. J. Rizvi, W. Aslam, M. Shahzad, S. Saleem, and M. M. Fraz, PROUD-MAL: Static analysis-based progressive framework for deep unsupervised malware classification of Windows portable executable, Complex & Intelligent Systems, (2022) 1–13.
[34] S. Ritwika and K. B. Raju, Malicious software detection and analyzation using the various machine learning algorithms, in: 2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT), IEEE, 2022, pp. 1–7.
[35] I. Obeidat and M. AlZubi, Developing a faster pattern matching algorithm for intrusion detection system," International Journal of Computing, 18(3) (2019) 278–284.
[36] E. M. Cherrat, R. Alaoui, and H. Bouzahir, Score fusion of finger vein and face for human recognition based on convolutional neural network model, International Journal of Computing, 19(1) (2020) 11–19. doi:10.47839/ijc.19.1.1688.
[37] N. Kayhan, S. Fekri-Ershad, Content based image retrieval based on weighted fusion of texture and color features derived from modified local binary patterns and local neighborhood difference patterns, Multimedia Tools and Applications, 80(21) (2021) 32763- 32790.