Methods for detecting software implants in corporate networks

Dmytro Denysiuk[1,†], Tomas Sochor[2,†], Mariia Kapustian[1,*,†], Antonina Kashtalian[1,†] and Oleg Savenko[1,†]

[1] *Khmelnytskyi National University, Instytutska str., 11, Khmelnytskyi, 29016, Ukraine*
[2] *Prigo University, Havirov, Czech Republic*

Abstract

With innovations in the technological sphere, the development of mechanisms that allow obtaining confidential information without the proper authorization of the owner is increasing. One of such mechanisms is software implants. This type of software is very difficult to detect because it does not use specialized signatures or code obfuscation, making it difficult to detect. This paper proposes a software implant detection system based on recurrent neural networks and a classifier. The classifier is a mechanism that describes the operating behavior of the software and provides the recurrent neural network with the ability to learn. This mechanism helps to identify behavioral patterns characteristic of software implants and notify the user of the possible risk of data loss.

During the experiments, it was found that in order to successfully detect a software implant that initiates the creation of additional processes, the system needs to be trained for 50 epochs. Thus, the detection efficiency is 97.50%, which indicates the possibility of using this system as an effective mechanism for detecting software implants in corporate systems. Given the results obtained, it can be recommended for use in a wide range of information systems to ensure reliable protection against potential security threats.

Keywords

malware detection, distributed computing, heterogenous computer systems, decoys, software implants, deception system

1. Introduction

Data security is particularly important in today's world, where society and the economy are becoming increasingly digitalized and interconnected. The growing volume of electronic data and its strategic importance in various fields of activity necessitates a comprehensive approach to ensuring the security, integrity and availability of this data. Modern technological innovations, such as cloud computing, the Internet of Things (IoT) and

IntelITSIS'2024: 5th International Workshop on Intelligent Information Technologies and Systems of Information Security, March 28, 2024, Khmelnytskyi, Ukraine

* Corresponding author.
† These authors contributed equally.
denysiuk@khmnu.edu.ua (D. Denysiuk); tomas.sochor@osu.cz (T. Sochor); kapustianm@khmnu.edu.ua (M. Kapustian), yantonina@ukr.net (A. Kashtalian); savenko_oleg_st@ukr.net (O. Savenko)
0000-0002-7345-8341 (D. Denysiuk); 0000-0002-1704-1883 (T. Sochor); 0000-0001-9200-1622 (M. Kapustian); 0000-0002-4925-9713 (A. Kashtalian); 0000-0002-4104-745X (O. Savenko)

artificial intelligence, require stricter security measures as they open up new opportunities for abuse and exploitation. Ensuring effective protection against potential threats requires research into innovative methods of data protection and detection. Particular attention should be paid to the interaction between entities that process and store data and entities responsible for software development. Such an integrated approach may take into account technical, organizational and legal aspects of data protection. Ensuring data security in the modern world is a highly specialized task that requires constant improvement and adaptation to unpredictable threats and challenges in the ever-changing digital environment.

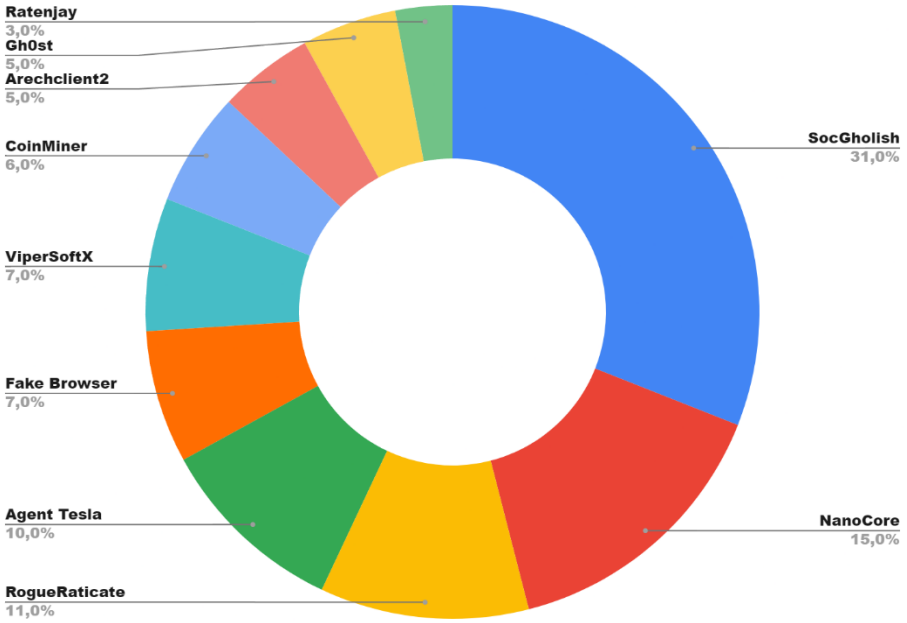

Figure 1: Analysis of the most malicious software.

SocGholish [2] – is a malicious downloader implemented in the JavaScript language that spreads through malicious or compromised websites. It uses fake software updates, such as web browser or Flash updates, to trick users and even social engineering tools to make the entered requests look innocent. The malware uses various methods to redirect traffic and spread the payload. It is noted that SocGholish uses Cobalt Strike to steal information from victims' systems. In addition, SocGholish infection can cause other types of attacks, such as downloading the NetSupport Remote Access Tool, Async Remote Access Tool, and, in some cases, even ransomware.

NanoCore [3] – is a remote access tool (RAT[4]) that spreads via spam emails with an attachment such as a malicious Excel XLS spreadsheet. This malicious tool interacts with the system by receiving commands to download and execute files, visit websites, and create a RunOnce key in the victim's registry for later storage. Its spread and interaction with the system can cause serious consequences for computer security and data privacy on infected systems.

RogueRaticate[5] – is a malicious downloader implemented with JavaScript that spreads through vulnerable or compromised websites using fake browser updates. The RogueRaticate payload is represented by an HTML application file that is archived or downloaded as a shortcut file. RogueRaticate infection leads to further attacks, such as downloading legitimate remote access tools, such as NetSupport, using the attackers' tools.

Agent Tesla[6] – is a malware designed for Windows operating systems. It can be purchased on criminal forums as a Malware-as-a-Service[6] (MaaS). Depending on the version purchased, it has a variety of functionalities, such as intercepting keystrokes and screenshots, collecting saved credentials from web browsers, copying the contents of the clipboard, invading the victim's files, and downloading other malware to the computer. These capabilities allow the attacker to carry out various attacks and criminal activities, gaining control over the victim's system.

Fake Browser[5] – is a downloader virus that uses the JavaScript language and spreads through malicious or compromised websites via fake browser updates. Note that Fake Browser infection may result in additional infected actions, such as using a remote access tool such as NetSupport. This opens up the possibility for the virus to perform various criminal actions and further spread the malicious impact to the targeted systems.

ViperSoftX[7] – is a cryptocurrency thief with multiple levels of complexity. It is usually distributed in the form of a malicious crack for known software via torrent networks and file sharing sites. This method of distribution often facilitates its unnoticeable penetration and activation on the systems of users using illegal copies of the software. This opens up the possibility for ViperSoftX to use its advanced capabilities to engage in cryptocurrency mining, launch attacks and other malicious activities that have a major impact on infected systems.

CoinMiner[8] – are a family of cryptocurrency miners that primarily use Windows Management Instrumentation (WMI) to distribute themselves across the network. Additionally, they often use the standard WMI event handler to execute scripts and ensure constant activity. However, the functionality of this malware can vary as there are different variants within the family. CoinMiner is often distributed via spam emails or in combination with other malware.

Arechclient2[9] – is a network virus, also known as SectopRAT, with a variety of functionalities, including the ability to bypass security systems. This malicious tool is capable of analyzing victims' systems, stealing sensitive information such as browser data and cryptocurrency wallets, and initiating the creation of a hidden desktop to control browser sessions. Moreover, it is equipped with a number of features aimed at overcoming the protection of virtual machines and emulators

Gh0st[10] – is a remote administrative tool (RAT) used to control infected endpoints. By interacting with other malicious programs, Gh0st creates a secret channel for the attacker to communicate. This approach opens up the possibility of unauthorized control and various attacks.

Ratenjay[11] – is a remote control tool that infects a victim's system by coinciding with other malware or during file uploads. Once infected, it activates remote command execution and is equipped with a keylogging feature that allows the attacker to monitor keystrokes on

the victim's system. This functionality allows the attacker to gain access to confidential information entered by the user in order to further use it for criminal purposes.

Much of the software provided can be defined as a software implant. This indicates that these programs are embedded in the system in order to create remote access mechanisms and perform various functions without the user's knowledge, which can have serious consequences for the security and confidentiality of information. Software implants differ from other types of malware in that they are more difficult to detect, especially because they differ slightly from standard program code. Additionally, parts of a software implant can be embedded in normal functions of a software product, making them invisible to normal monitoring and analysis. The importance of analyzing this feature is that implants mask their presence by embedding themselves in the normal circulation of program code. This ability reduces the possibility of detecting software implants and requires the development of highly effective methods for detecting such malicious elements.

Due to the unique nature of software implants, it is important that research focuses on their identification and the development of innovative analysis tools. One area of research is to develop a mechanism for detecting software implants in software products. A software implant, which is a piece of code created by a developer to gain unauthorized access, is a focus of attention in efforts to improve the security and protection of information systems. The development of effective tools to detect these implants is defined as a step towards improving information security and addressing potential vulnerabilities to cyberattacks. Consideration of the mechanisms of implementation of software implants and their impact on the functioning of software products expands the understanding of challenges in the field of system protection and contributes to the development of strategies to counter these threats. Analyzing the characteristics of implants is important to identify their potential impact, which will provide more effective means of combating these new forms of attacks on the information system. Thus, the class of malware under consideration differs from the other classes in that it includes malware that does not reproduce or transfer on its own, but is embedded in useful software at the stage of developing it. Therefore, it is difficult to detect and requires the development of new and improvement of existing detection methods. Thus, there is a need to develop methodologies for detecting software implants based on mechanisms for analyzing the behavior of system components in order to detect software implants at the verification stage of a software product before its implementation in the IT infrastructure.

In this paper, we propose to use a mechanism for detecting software implants based on the analysis of the behavior of system resources relative to the expected behavior of the program, which is set through a classifier.

2. Characteristics of software implants in related works

The main characteristics of software implants that distinguish them from other classes of malware are presented in modern related works:

No code obfuscation [12]. Software implants differ from other types of malware in that their code is not subject to obfuscation[19,20]. Obfuscation is the process of transforming or hiding program code to make it difficult to understand and analyze.

Lack of signatures [13]. Due to the fact that a software implant is actually embedded in the software, it does not have suspicious signatures that are standardly used by antivirus programs to detect malware. Thus, software implants successfully avoid detection by antiviruses during scans.

Lack of direct commands in the source code of the software implant. In this context, there are no direct instructions that can be executed in the specified software file. This property provides resistance to detection during visual inspection of the source code.

The absence of direct connections with developers is an important feature of software implants. They avoid direct connections with developers to avoid detection and intervention by antivirus systems. This mechanism not only guarantees anonymity, but also reduces the risk of compromising the developer of the software implant.

The ability to receive commands from external software resources is a key feature of software implants. Software implants can use external files to receive commands to execute. For example, these commands can be encrypted using steganography techniques[14] and embedded in images. When the software implant is hosted on a web server, the image files can be periodically updated to allow for new commands. This approach allows the software implant to be controlled without undue detection through additional network connections.

An important step in developing an effective malware implant detection method is the classification stage by attack method. This classification includes ransomware, remote access programs, programs that wait for commands, and programs that collect and transmit information. Further study and analysis of these categories allows you to create reliable detection methods for various types of software implants.

Ransomware - is a category of software implants aimed at extorting and collecting confidential information related to a user or system. It initiates the process of extorting this information by using unauthorized methods to gain access and interfere with the system. This type of implant is aimed at obtaining confidential data for further use for profit.

A program waiting for the team, is a class of implants designed to execute specific commands or instructions coming from an attacker. Its main purpose is to perform specific tasks or attacks using the received remote commands.

A program that provides remote access, is a category of implants aimed at creating remote access points to a system. This allows attackers to perform remote operations and manipulate system resources.

Thus, the classification of software implants according to certain features and criteria is the basis for the development of new and improvement of the known methods of their detection.

3. Intrusion detection methods

Software implants that have similarities to Trojan horses in their purpose pose a potential threat to systems. In order to effectively detect and prevent their introduction and activities, various mechanisms are used, which can be divided into detection mechanisms and prevention mechanisms.

The detection mechanisms include obfuscation analysis[19]. Obfuscation analysis is a process of systematic study and analysis of the obfuscation techniques used in program

code. In this context, obfuscation refers to techniques aimed at making it difficult to read and understand program code by introducing various transformations, transformations and hiding program logic. Obfuscation analysis involves a detailed study of the obfuscation techniques used to determine the degree of difficulty in revealing the true functionality of the program. This process includes identifying and deciphering various obfuscation techniques, determining the level of code complexity and its impact on the readability and understanding of the program.

In order to analyze and detect code obfuscation, we consider a methodology aimed at identifying metamorphic viruses. This approach is based on a thorough study of obfuscation features that characterize program code modifications. The focus is on identifying and analyzing variations in the functional blocks of suspicious programs and their modified versions. This strategy is aimed at identifying certain obfuscation patterns that are characteristic of metamorphic viruses and allows to increase the efficiency of the process of detecting such threats. Therefore, an important step in the process of detecting a software implant in software is to systematically check the program code to identify the presence of obfuscation patterns. This stage of analysis is aimed at identifying changes made to the code by the implant that may hide its existence or complicate the detection of malicious actions. The emphasis is placed on identifying characteristic obfuscation patterns and analyzing them to effectively detect potentially malicious modifications in the program code.

One approach to malware detection within distributed systems with partial centralization[18]. This method involves the use of distributed systems operating with partial centralization, which include decentralized subsystems where decisions on malware detection are made by a centralized management body. To determine the presence of malware in the system components, characteristic indicators have been established, and generalized analytical expressions for their calculation have been developed. These analytical expressions make it possible to assess the state of system components and make appropriate decisions to optimize the malware detection process.

Detection mechanisms include an expanded range of technical and analytical tools designed to detect and analyze software implants. These mechanisms are based on a thorough analysis of program code, detection of anomalies during program execution, monitoring of system resources, security audits, and the use of static and dynamic analysis to identify potentially harmful elements. Additionally, detection mechanisms include measures aimed at protecting against code modification, detecting illegal connections to the developer, and effectively monitoring and analyzing the interaction of the program with system resources. The interaction of these mechanisms helps to detect software implants and provides a high level of protection of information systems from potential threats.

Prevention mechanisms include a variety of technical and organizational strategies designed to make it impossible to introduce and effectively control software implants. These mechanisms are focused on detecting and deviating from standard behavior that may indicate the presence of malicious program code.

Prevention systems include the effective implementation of intrusion detection systems (IDS[15]) to respond to anomalies in network traffic in a timely manner. In addition, it is important to implement and configure mechanisms for tracking program behavior that allow you to actively recognize abnormal changes and activities in the program code and

operating system. For the proper functioning of IDSs, an important step is a detailed analysis of their work, which can be presented in the form of a scheme, as shown in Fig. 2. 2. This scheme allows the intrusion detection system to effectively analyze network traffic, detect abnormal patterns, and take appropriate measures to prevent further incidents. The approach of using intrusive detection systems and mechanisms for tracking program behavior provides comprehensive protection of information systems from potential threats, ensuring the detection and neutralization of possible intrusions.

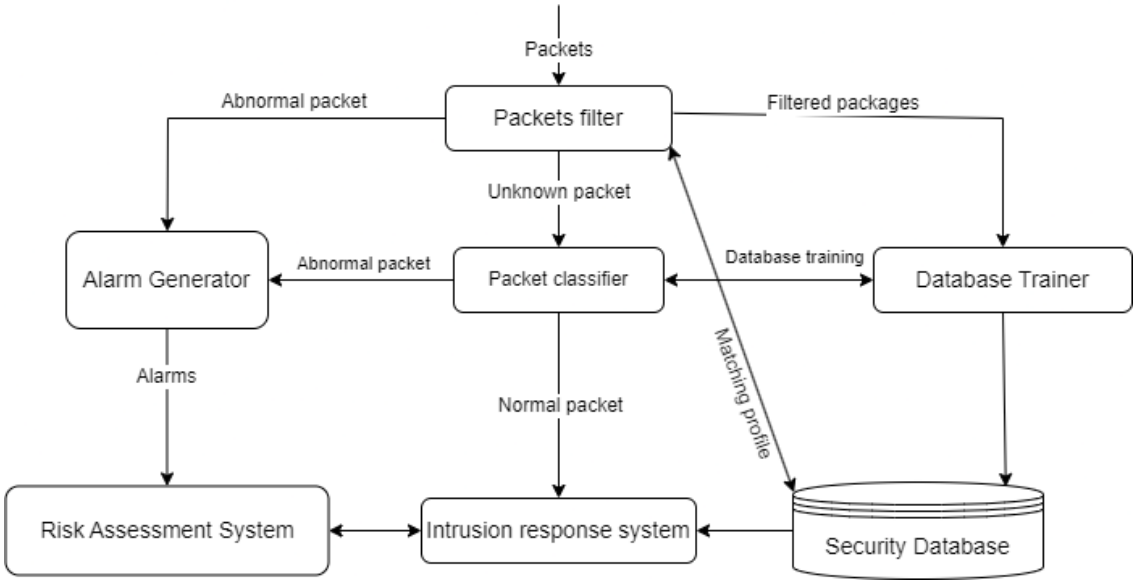

Figure 2: Intelligent intrusion detection scheme based on anomalies.

Anomaly Behavior Detection (ABD) systems[16] are a tool for detecting deviations in the behavior of a system or user from a predictable pattern. They are used in a variety of areas, including intrusion detection, fraud prevention, defect detection, system performance monitoring, event detection in network sensors, and ecosystem disruption response.

ABD systems fall into two main categories:

1. Misuse Detection Systems (MDS): These systems respond to known attacks using signatures and patterns of behavior that have previously been classified as abuse.
2. Anomaly Detection Systems (ADS): These systems record deviations from the normal development of a system, not necessarily predefined. They detect unusual patterns or anomalies that may indicate potential threats.

An essential characteristic of ABD systems is their ability to handle attacks even when the attacker has already gone through the authentication and authorization process and the formal access rights correspond to his or her authority. This makes it possible to detect anomalies that occur during legitimate operations but deviate from the typical context of behavior. The structure of the process with the depicted scheme is shown in Fig. 3. This scheme demonstrates the operation of the anomaly detection system algorithm, where each

block corresponds to a specific stage of analysis and decision-making. Such an anomaly detection system allows to effectively identify unforeseen deviations in behavior, enhancing the overall level of protection of information systems from potential threats and attacks.

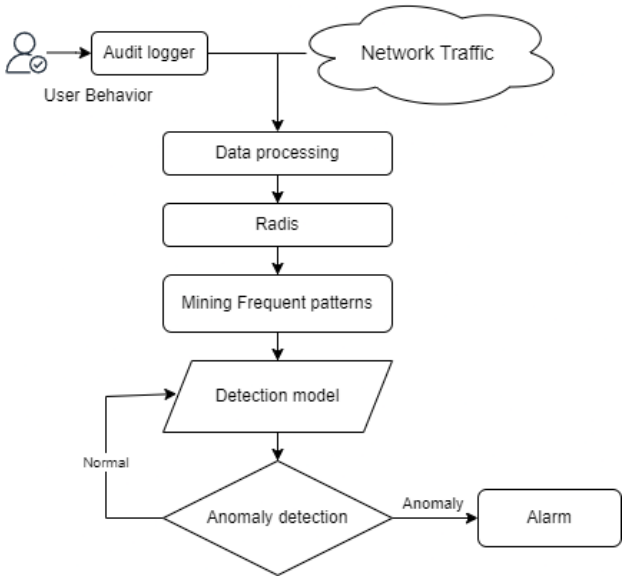

Figure 3: Scheme of the system for detecting abnormal behavior.

Thus, there is a need to develop a system for detecting software implants in order to improve the effectiveness of information system protection. This initiative is driven by the growth of threats and the emergence of new methods of attacking the digital environment. The development of such a system is determined by the need to reliably detect and analyze software implants that are difficult to detect and differ from regular program code. The creation of this system involves consideration of innovative methods aimed at effectively detecting and protecting against potential digital security threats, ensuring reliable control over digital Front matter.

4. Method for detecting software implants using a decoy system

A decoy method can be used to detect software implants. This system will emulate the operation of the subsystem, simultaneously analyzing requests to program files, network requests from the program, and process interaction. It will also perform the functions of creating, running, restarting and monitoring data that is loaded into the system's memory. The described decoy system is defined by its ability to emulate the behavior of a typical object that can be targeted by software implants. It performs the functions of analyzing and observing actions that may indicate a potential threat or inappropriate interference in the system. The work of the algorithm is shown in Fig. 4. This diagram allows you to understand in detail the sequence of operations performed by the decoy system to detect possible software implants.

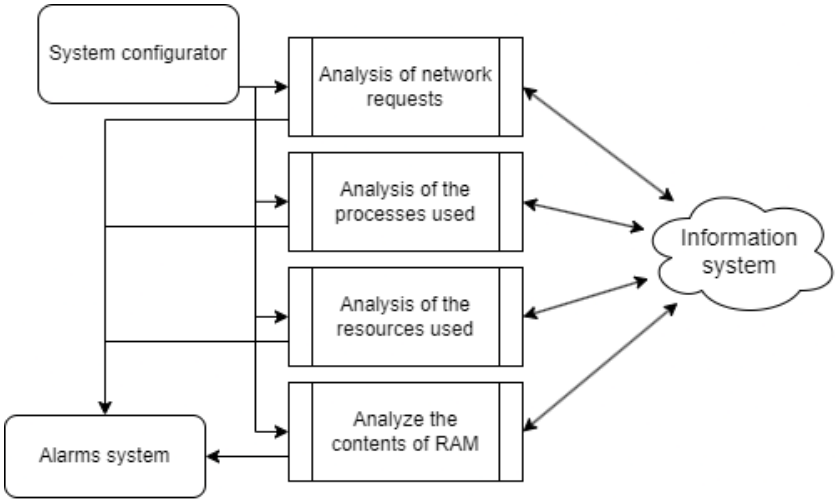

Figure 4: Concept of a software implant detection system.

Since a software implant is a piece of code that is virtually indistinguishable from regular code, its effective detection requires that the initial settings be determined using a system configurator. This configurator is a tool that includes a set of patterns that characterize a given software product. These patterns combine information about the program's interaction with resources, a description of TCP/IP, the interaction of processes performed by the software, and the structure of the RAM used by the software.

The system configurator acts as a system tool, setting the parameters and characteristics of the software in accordance with the specified patterns. This process provides the algorithm with the ability to detect and respond to abnormal parameters in the software operation. Due to the difficulties of detecting software implants, the use of a system configurator becomes a necessary element for customizing the detection system. This allows the system to be adapted to different types of software, providing the ability to effectively respond to a variety of implant threats.

Network request analysis includes a network traffic scanning mechanism that actively evaluates whether the software generates additional requests to third-party resources, taking into account the resources that were described in the configurator. To effectively detect software implants, the system analyzes the frequency of requests, their number, and the length of packets in each request. Additionally, the system checks the content of requests that were not specified in the configurator. This makes it possible to determine whether incoming requests contain external commands and whether outgoing requests contain confidential information.

To analyze data in network queries, a recurrent neural network (RNN) is used[17]. Due to the characteristics of recurrent neural networks, traffic analysis and anomaly detection in the network data stream are efficient. These features of recurrent neural networks contribute to more accurate and sensitive detection of anomalous data in network traffic, which makes this method important for identifying potential threats and inconsistencies in network behavior. The principle of operation of the network query analysis mechanism is shown in Fig. 5.

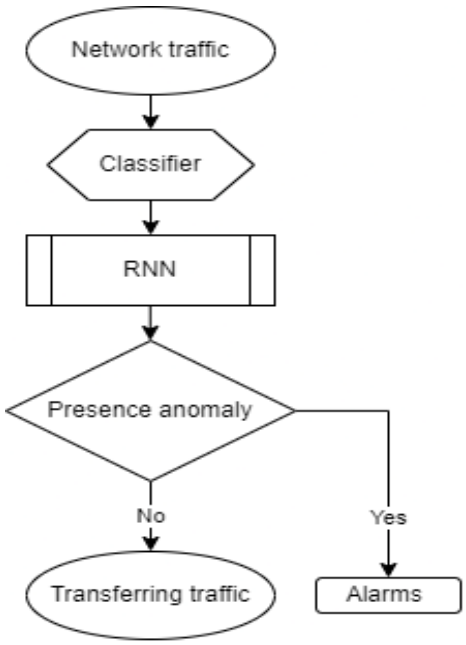

Figure 5: Scheme for analyzing network requests.

To improve the efficiency of detecting software implants aimed at creating botnet networks, a multi-agent approach in computer systems can be applied [21-23]. This approach can serve as an additional stage of system analysis. The multi-agent approach to detecting botnets in computer systems is a methodology based on the concept of multi-agent systems to detect and counteract botnets. A botnet is defined as a network of computers or other devices that are controlled by an attacker and used to perform various tasks, such as spreading malware or attacking other systems. The multi-agent approach involves the use of autonomous agents that act independently and interact with each other to detect and counter botnets. Each agent can perform its own functions and tasks, such as analyzing network traffic, detecting anomalies, or analyzing the characteristics of the attacks used. This approach is aimed at improving systems for detecting and responding to threats posed by botnets, providing a more efficient and adaptive mechanism for protecting computer systems.

The next stage of detecting software implants is carried out by monitoring the processes generated by the software. This mechanism operates with a table of allowed processes and procedures to determine the occurrence of anomalies in processes. It includes a task manager for active software and a mechanism for detecting new resources. When a new process is detected, the system stops its operation and checks whether it can be re-created. If the process is restored, the system displays a message about the probability of the presence of a software implant that initiates the restoration of processes. The application of this method is based on the study and analysis of process dynamics, relying on the principles of monitoring and analyzing changes in system resources to timely detect and counteract cases of creating unforeseen processes. The operation of the system for detecting abnormal processes is shown in Fig. 6.

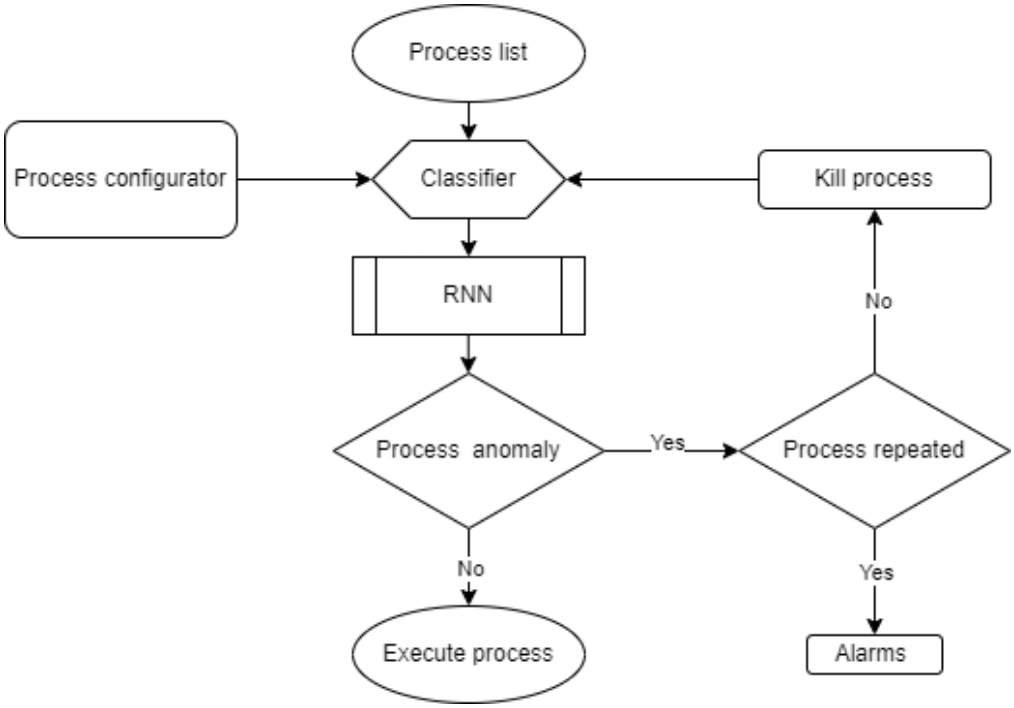

Figure 6: Algorithm of the system for detecting abnormal processes.

This algorithm performs a number of key functions. First, it checks whether the process belongs to the list of configured processes. Next, it checks the classification of the process and analyzes the resources it uses. A recurrent neural network (RNN[24-25]) is used to assess possible threats. If the process is suspicious, the system tries to stop its operation. If the suspicious process is repeated, the system generates information about its potential danger to the system. This approach is based on the use of scientific methods and takes into account aspects of process behavior analysis in order to effectively identify and eliminate possible threats.

The algorithm for analyzing used resources and RAM content operates in accordance with the principles used in the algorithm for analyzing network requests. When analyzing the contents of the RAM, the system compares the declared amount of allocated memory described in the classifier with the actual amount of memory allocated. As for the analysis of used resources, the system examines which files are used in the program and how they are used. For example, if it is found that a file that is not intended to read data in binary format at all, such as an image, is processed as a binary file, this may indicate an attempt to manipulate the system through file input.

Thus, the described method can be implemented and the detection strategy set by the steps of this method is promising for further research.

5. Experiment

Setting up the experiment. The experiment involves the use of the developed system, which implements the method of detecting software implants. The conditions of the

experiment were clearly defined. It is noted that the application of this method is possible in the corporate environment, since it is in corporate information systems that software implants can mainly occur. The experiment involved software implants aimed at various tasks:

- Collecting information and transferring it to the server.
- The implant waiting for the command is hidden in the image using steganography.
- A program that provided access to the system console.
- Implants designed to create fake accounts.

For the experiment, a recurrent neural network was used to classify software implants depending on their functionality:

1. Interaction with the network.
2. Operations with files.
3. Process management.
4. Execute commands from the command line.

The object for classification was a system specially designed for warehouse management. The classifier included characteristics related to the types of files that the software product processed, as well as a detailed description of the operations that could be performed on the files. The servers to which the program made requests and the console operations available to the software implant were also identified.

To objectively evaluate the algorithm, more than 3,000 different operations were performed during the experiment, including 400 operations that were specific to software implants. This had a positive impact on the quality of classification of software implants and allowed for a deeper understanding of their functional purpose within the experiment.

Results of experimental studies. To perform the experiment, an isolated system and a system that simulated user actions were used. In order to effectively train the recurrent neural network and correctly classify software implants, it was planned to spend 50 epochs training the model.

The results of the experiment are presented in Table 1, which shows the data for every 10 epochs. This allowed us to obtain detailed data on the detection and classification of software implants during the experiment. The results of the experiment are shown in Table 1.

During the experiment, it was found that the network was effectively trained for 50 epochs. It was noted that the network demonstrated the highest efficiency when detecting software implants that initiate additional processes.

The overall efficiency of detecting software implants that create additional processes was 98.00%. The training schedule of the software implant detection system is shown in Fig. 7.

Table 1

Results of the experiments, TP - True positive, TN - True negative, FN - False positive, FP - False negative.

Epochs of learning	Classes of implants	TP	TN	FN	FP	Overall accuracy, %
1-10	1	102	100	102	96	50.50%
	2	101	85	110	104	46.50%
	3	111	95	105	89	51.50%
	4	111	95	99	95	51.50%
10-20	1	143	123	70	64	66.50%
	2	125	134	69	72	64.75%
	3	120	155	70	55	68.75%
	4	125	132	69	74	64.25%
20-30	1	137	137	55	71	68.50%
	2	131	150	44	75	70.25%
	3	143	125	73	59	67.00%
	4	134	124	73	69	64.50%
30-40	1	156	161	47	36	79.25%
	2	139	168	48	45	76.75%
	3	155	168	48	46	78.00%
	4	159	160	47	34	79.75%
40-50	1	218	172	4	6	97.50%
	2	179	211	5	5	97.50%
	3	191	201	3	5	98.00%
	4	206	185	3	6	97.75%

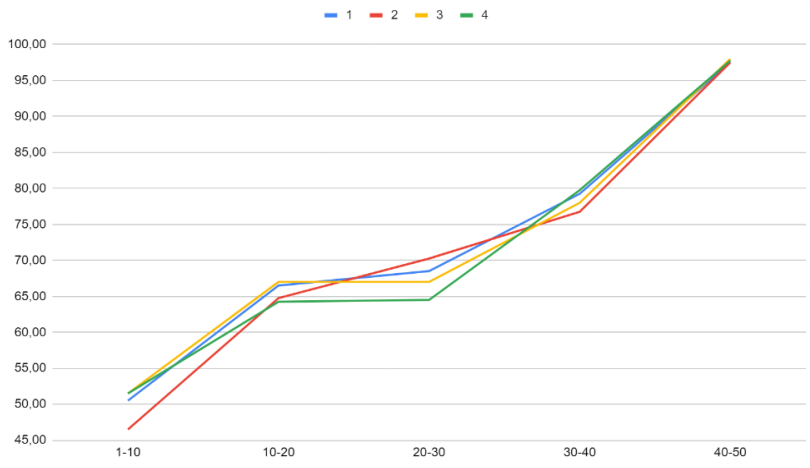

Figure 7: System learning dynamics.

6. Conclusions

As a result of the study, a software implant detection system based on recurrent neural networks and a classifier was developed. The classifier is a mechanism that describes the operating behavior of the software and provides the recurrent neural network with the ability to learn. This mechanism helps to identify behavioral patterns characteristic of software implants and notify the user of the possible risk of data loss.

During the experiments, it was found that in order to successfully detect a software implant that initiates the creation of additional processes, the system needs to be trained for 50 epochs. Thus, the detection efficiency is 97.50%, which indicates the possibility of using this system as an effective mechanism for detecting software implants in corporate systems. Given the results obtained, it can be recommended for use in a wide range of information systems to ensure reliable protection against potential security threats.

Further research work is aimed at creating automatic classification systems designed to automate the process of setting up a software implant detection system. This will help reduce the time required to configure and put into operation a system designed to detect software implants in various types of software and information systems.

References

[1] Top 10 Malware Q3, 2023. URL: https://www.cisecurity.org/insights/blog/top-10-malware-q3-2023

[2] S. Poudyal, D. Dasgupta, AI-powered ransomware detection framework, IEEE Symposium Series on Computational Intelligence (SSCI). (2020) 1154-1161.

[3] V. Valeros, S. Garcia, Growth and commoditization of remote access Trojans, IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). (2020) 454-462.

[4] O. Ude, B. Swar, Securing Remote Access Networks using malware detection tools for industrial control systems. 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS). (2021) 166-171.

[5] Enterprise Cybersecurity Solutions, Services & Training | Proofpoint US, 2024. URL: https://www.proofpoint.com/us/blog/threat-insight/are-you-sure-your-browser-date-current-landscape-fake-browser-updates

[6] G.F.M. Karo-Karo, M.S.A. Harumnanda, C. Lim, Investigating Multiple Malware as a Service (MaaS): Analysis and Prevention Techniques. IEEE International Conference on Cryptography, Informatics, and Cybersecurity (ICoCICs). (2023) 270-274.

[7] H. Shuang, L. Zhao, D. Lie, vWitness: Certifying Web Page Interactions with Computer Vision, 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 2023, pp. 431-444. URL: https://security.csl.toronto.edu/blog/wp-publications/hshuangdsn2023vwitness

[8] Sudhakar, S. Kumar, An emerging threat Fileless malware: a survey and research challenges. Cybersecurity, vol. 3 (2020) 1-11.

[9] MalwareBazaar | Browse Checking your browser, 2024. URL: https://bazaar.abuse.ch/browse/tag/Arechclient2/

[10] Y. Zhang, H. Xue, J. Lin, X. Liu, W. Gai, X. Yang, A. Wang, Y. Yue, B. Sun, ER-ERT: A Method of Ensemble Representation Learning of Encrypted RAT Traffic. 2023 IFIP Networking Conference (IFIP Networking) (2023) 1-10.
[11] J. Ježovnik, , Glasovne in naglasne značilnosi terskega narečja. Založba ZRC, vol. 43 (2022).
[12] S. Zafar, M.U. Sarwar, S. Salem, M.Z. Malik, Language and obfuscation oblivious source code authorship attribution. IEEE Access (2020) vol. 8, 197581-197596.
[13] J. Acharya, A. Chaudhary, A. Chhabria, S. Jangale, Detecting malware, malicious URLs and virus using machine learning and signature matching. 2021 2nd International Conference for Emerging Technology (INCET) (2021) 1-5.
[14] D. Denysiuk, O. Savenko, S. Lysenko, B. Savenko and A. Kashtalian, Method for Detecting Steganographic Changes in Images Using Machine Learning, 2023 13th International Conference on Dependable Systems, Services and Technologies (DESSERT), Athens, Greece, (2023) 1-6, doi: 10.1109/DESSERT61349.2023.10416453.
[15] M. Eskandari, Z.H. Janjua, M. Vecchio, F. Antonelli, Passban IDS: An intelligent anomaly-based intrusion detection system for IoT edge devices. IEEE Internet of Things Journal, (2020) vol. 7, 6882-6897.
[16] H. Zhang, J. Li, X. Liu, C. Dong, Multi-dimensional feature fusion and stacking ensemble mechanism for network intrusion detection. Future Generation Computer Systems, (2021) vol. 122, 130-143.
[17] K. Guo, Y. Hu, Z. Qian, H. Liu, K. Zhang, Y. Sun, J. Gao, B. Yin, Optimized graph convolution recurrent neural network for traffic prediction. IEEE Transactions on Intelligent Transportation Systems (2020) vol. 22, 1138-1149.
[18] B. Savenko, A. Kashtalian, S. Lysenko and O. Savenko, Malware Detection By Distributed Systems with Partial Centralization, 2023 IEEE 12th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Dortmund, Germany, 2023 pp. 265-270
[19] S.S. Albouq, A.A. Abi Sen, A. Namoun, N.M. Bahbouh, A.B. Alkhodre, A. Alshanqiti, A double obfuscation approach for protecting the privacy of IoT location based applications. IEEE Access (2020) vol. 8, 129415-129431.
[20] 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.
[21] Y. Tesnim, J. Farah, A multi-agent-based system for intrusion detection. Agents and Multi-Agent Systems: Technologies and Applications 2021: Proceedings of 15th KES International Conference, KES-AMSTA 2021, June 2021 (2021) 177-191.
[22] C. Liang, B. Shanmugam, S. Azam, A. Karim, A. Islam, M. Zamani, S. Kavianpour, N.B. Idris, Intrusion detection system for the internet of things based on blockchain and multi-agent systems. Electronics (2020) vol. 9, 1120.
[23] S.R. Zahra, M.A. Chishti, A generic and lightweight security mechanism for detecting malicious behavior in the uncertain Internet of Things using fuzzy logic-and fog-based approach. Neural Computing and Applications (2022) vol. 34, 6927-6952.
[24] A. Tsantekidis, N. Passalis, A. Tefas, Recurrent neural networks. Deep learning for robot perception and cognition, Elsevier (2022) 101-115.
[25] L. Ruiz, F. Gama, A. Ribeiro, Gated graph recurrent neural networks. IEEE Transactions on Signal Processing (2020) vol. 68, 6303-6318.