Edited by
Oleh Harasymchuk

MODERN METHODS OF ENSURING INFORMATION PROTECTION IN CYBERSECURITY SYSTEMS USING ARTIFICIAL INTELLIGENCE AND BLOCKCHAIN TECHNOLOGY

Monograph

Technology® Center

2025

UDC 004.056
M78

Published in 2025
by TECHNOLOGY CENTER PC®
Shatylova dacha str., 4, Kharkiv, Ukraine, 61165

M78
Authors:
Edited by Oleh Harasymchuk
Ivan Opirskyy, Oleh Harasymchuk, Olha Partyka, Vitalii Susukailo, Andrian Piskozub, Dmytro Sabodashko, Sviatoslav Vasylyshyn, Anatoliy Obshta, Yevhenii Kurii, Danyil Zhuravchak, Ivan Tyshyk, Andrii Partyka, Petro Haraniuk, Taras Kret, Volodymyr Yuzevych, Viktor Otenko, Yuriy Nakonechnyy, Nazarii Dzianyi, Leonid Bortnik, Marta Stakhiv, Taras Lukovskyy, Andriy Horpenyuk, Stepan Voytusik, Roman Kuten, Ivan Kolbasynskyi, Khrystyna Besaha, Yurii Furdas, Oleksandr Isakov, Roman Andriiv, Oleh Tsebak
Modern methods of ensuring information protection in cybersecurity systems using artificial intelligence and Blockchain technology: monograph / O. Harasymchuk and others. – Kharkiv: TECHNOLOGY CENTER PC, 2025. – 132 p.

The monograph discusses the methodology for cooperative conflict interaction modeling of security system agents. The concept of modeling the structure and functioning of the security system of critical infrastructure facilities is demonstrated. The method for assessing forecast of social impact in regional communities is presented. Counteracting the strategic manipulation of public opinion in decision-making by actors of social networking services based on the conceptual model for managed self-organization in social networking services are developed. Algorithms for thinning the critical infrastructure identification system and their software are implemented.
The monograph is intended for teachers, researchers and engineering staff in the field of cybersecurity, information technology, social engineering, communication systems, computer technology, automated control systems and economic information security, as well as for adjuncts, graduate students and senior students of relevant specialties.
Figures 59, Tables 18, References 169 items.

This book contains information obtained from authentic and highly regarded sources. Reasonable efforts have been made to publish reliable data and information, but the author and publisher cannot assume responsibility for the validity of all materials or the consequences of their use. The authors and publishers have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission to publish in this form has not been obtained. If any copyright material has not been acknowledged please write and let us know so we may rectify in any future reprint.
The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made.
Trademark Notice: product or corporate names may be trademarks or registered trademarks, and are used only for identification and explanation without intent to infringe.

DOI: 10.15587/978-617-8360-12-2
ISBN 978-617-8360-12-2 (on-line)

Cite as: Harasymchuk, O. (Ed.) (2025). Modern methods of ensuring information protection in cybersecurity systems using artificial intelligence and Blockchain technology: collective monograph. Kharkiv: TECHNOLOGY CENTER PC, 132. doi: http://doi.org/10.15587/978-617-8360-12-2

9 786178 360122

Copyright © Author(s) 2025
This is an open access paper under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (CC BY-NC-ND 4.0)

AUTHORS

IVAN OPIRSKYY
Doctor of Technical Sciences, Professor, Head of Department
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0002-8461-8996

OLEH HARASYMCHUK
PhD, Associate Professor
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0002-8742-8872

OLHA PARTYKA
PhD, Associate Professor
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0002-3086-3160

VITALII SUSUKAILO
PhD, Assistant
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0003-4431-9964

ANDRIAN PISKOZUB
PhD, Associate Professor
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0002-3582-2835

DMYTRO SABODASHKO
PhD, Senior Lecturer
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0003-1675-0976

SVIATOSLAV VASYLYSHYN
PhD, Associate Professor
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0003-1944-2979

ANATOLIY OBSHTA
Doctor of Technical Sciences, Professor
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0001-5151-312X

YEVHENII KURII
PhD, Assistant
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0002-3423-5655

DANYIL ZHURAVCHAK
Assistant
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0003-4989-0203

IVAN TYSHYK
PhD, Associate Professor
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0003-1465-5342

ANDRII PARTYKA
PhD, Associate Professor
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0003-3037-8373

PETRO HARANIUK
PhD, Associate Professor
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0002-7450-8881

TARAS KRET
Assistant
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0002-6333-3190

VOLODYMYR YUZEVYCH
Doctor of Physical and Mathematical Sciences, Professor
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0001-5244-1850

VIKTOR OTENKO
PhD, Associate Professor
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0003-4781-7766

YURIY NAKONECHNYY
PhD, Associate Professor
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0002-6046-6190

NAZARII DZIANYI
PhD, Associate Professor
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0001-9101-3701

LEONID BORTNIK
PhD, Senior Lecturer
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0002-6116-7491

MARTA STAKHIV
PhD, Associate Professor
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0002-4094-2081

TARAS LUKOVSKYY
PhD, Associate Professor
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0009-0008-1652-8121

ANDRIY HORPENYUK
PhD, Associate Professor
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0001-5821-2186

STEPAN VOYTUSIK
PhD, Associate Professor
Department of Information Systems and Technologies
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0003-4234-3303

ROMAN KUTEN
PhD, Senior Lecturer
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0002-5688-2976

IVAN KOLBASYNSKYI
Assistant
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0009-0008-0755-1039

KHRYSTYNA BESAHA
PhD, Assistant
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0009-0004-4294-6133

YURII FURDAS
Postgraduate student
Department "Software Engineering"
Ukrainian National Forestry University
ORCID ID: https://orcid.org/0009-0000-5789-9174

OLEKSANDR ISAKOV
Assistant
Department of Information Technology Security
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0009-0007-4632-9492

ROMAN ANDRIIV
Teacher
Separate structural unit "Berezhany Professional College of NUBiP of Ukraine"
ORCID ID: https://orcid.org/0009-0001-2450-5439

OLEH TSEBAK
Assistant
Department of Information Technology Security
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0009-0004-7202-2855

ABSTRACT

This scientific work is devoted to the development and improvement of information protection methods to counteract unauthorized access to information activity facilities and their telecommunication networks. The conclusions of the study indicate the need to implement innovative solutions to increase the level of security in modern cyberspace.

An improved algorithm for determining and transferring node hosts in the Blockchain system is proposed, which uses floating hosts to increase the adaptability of the network to external attacks. This allows to automatically close ports during scanning, making it difficult for attackers to access the system and increasing the overall level of network protection.

A study of GPT models has shown their high efficiency in detecting cyberattacks on information activity facilities and their telecommunication networks. GPT 4.0 has demonstrated increased efficiency in processing and detecting various types of attacks compared to GPT 3.5, which provides faster response time and improves the overall level of security.

The developed method of collecting event logs from decoys based on Blockchain provides high fault tolerance and reliability of logs, which is critically important for protecting information objects and telecommunication networks. The decentralized nature of Blockchain prevents unauthorized editing of information, creating a reliable system for storing attack data.

The developed model of a dynamic system of active traps based on software decoys using Blockchain technology integrates decentralized and automatically updated attributes of traps. This increases the effectiveness of network protection, reduces the load on the infrastructure and the response time of services during attacks, which increases the channel throughput and data transfer rate.

The developed mathematical description of the calculation of dynamic attributes of software decoys takes into account the capabilities of Blockchain Solana, which made it possible to model and optimize the distribution of network resources. This increased the effectiveness of protection and ensured a quick response of services during external attacks.

The method of using Blockchain-based software decoys obtained in the work increases the resources required by the attacker to carry out an attack, which increases the response time of cybersecurity specialists. The use of dynamic Blockchain-based software decoys demonstrates better performance compared to static and other dynamic analogues, increasing the overall level of computer network security. The proposed cybercrime research system detects known attacks 31 % faster and is able to detect unknown attacks thanks to training the Isolation Forest model. The time for analyzing cybercrimes has been significantly reduced thanks to the use of the GPT model, which provides an effective and fast response to threats.

KEYWORDS

Concept of a multi-loop security system, socio-cyber-physical systems, post-quantum security mechanisms.

CONTENTS

List of Tables ix
List of Figures x
Circle of readers and scope of application xii

Introduction 1

1 Analysis of the problem of using software decoys and review of existing solutions for information activity objects 4
1.1 Analysis of the current state of research and publications 4
1.2 Overview of software decoys and their purposes 9
1.2.1 Analysis of advantages and disadvantages of software decoys 11
1.2.2 Interaction levels in software decoys 12
1.2.3 Obstacles to the deployment of software decoys 13
1.3 The problem of using decoys and decoys to protect data in computer networks 14
1.4 Analysis of the use of Blockchain technology 20
1.4.1 Using Blockchain systems as security in real-world examples 22

2 Studying the use of blockchain properties to prevent unauthorized access to information activity facilities 24
2.1 Studying the possibilities of using blockchain to protect data in infrastructure facilities 24
2.2 The relationship between Blockchain technology and AI, Big Data and other advanced technologies 26
2.3 Security and privacy analysis 27
2.4 Advantages and disadvantages of using Blockchain technology to protect cyber systems and telecommunications networks 30

3 Research on the use of the properties of artificial intelligence models for detection of cyberattacks and analysis of cybercrimes on information activity objects 33
3.1 Issues of modern solutions for the study of cybercrimes of information systems 33
3.2 Using artificial intelligence algorithms to investigate information security events and incidents 41
3.3 Exploring the possibilities of using chatbots using the GPT model for event log analysis 46
3.4 Cybercrime research model concept 50
3.5 Using Blockchain technology to investigate cybercrimes at information activity facilities 52

4 Development of a method for using software decoys as elements of protection of computer networks of information activity objects based on blockchain technology 55
4.1 Development of a decoy-based model of decentralized communication 55
4.1.1 Description of host communication in the built blockchain network 56
4.1.2 Transformation of services during connection to nodes 58
4.2 Development of a method for a dynamic system built using software decoys 59

5 Research of the effectiveness of the method of using software decoys built on the basis of blockchain technology as elements of protection of information activity objects 64
5.1 Analysis of the security level of the system using software decoys based on the developed model and solutions for its protection 64
5.1.1 Description of the experiment: Sniffer attack 64
5.1.2 Description of the experiment: Scanning attack 65
5.1.3 Description of the experiment: DDoS attack 66
5.2 Experimental analysis of the security level during solution simulation 68
5.2.1 Study of the protection of the system using software decoys based on the developed model against sniffer attack 69
5.2.2 Study of the protection of the system using software decoys based on the developed model against scanning attack 70
5.2.3 Study of the protection of system using software decoys based on the developed model against DDoS attack 71
5.2.3.1 Analysis of network response time in systems using the developed model and its analogues 72
5.3 Comparative analysis of the developed dynamic method with static analogues 78

6 Development and research of the effectiveness of the cybercrimes research system model at information activity objects 80
6.1 Development of a threat research system model 80
6.2 Vulnerability management component 82
6.3 Development of a methodology for investigating cybercrimes based on anomaly detection using the Isolation Forest model and GPT, taking into account information system vulnerabilities and threat intelligence data 85
6.4 Practical implementation of the cybercrime investigation system model 89
6.5 Preparing a vulnerable environment for testing a cybercrime investigation system model 93
6.5.1 Training the Isolated Forest Model 95
6.5.2 Collection of additional information 96
6.6 Experimental attack execution 97
6.6.1 Experimental attack execution using automated scanning tools 97
6.6.2 Experimental injection attack 98

6.6.3 Experimental implementation of Directory Traversal attack....99
6.6.4 Experimental attempt to violate the logic of the program 100
6.7 Analysis of the results.... 101
6.7.1 Analysis of the speed of detection and investigation of attacks using automatic scanning tools.... 101
6.7.2 Analysis of the detection speed and investigation of injection attacks....102
6.7.3 Analysis of the speed of detection and investigation of Directory Traversal attacks.... 103
6.7.4 Analysis of the detection speed and investigation of attacks with violation of program logic 105
6.7.5 Analysis of the effectiveness of information security event detection by a cybercrime investigation system for information systems infrastructure components....106

Conclusions....107
References....109

LIST OF TABLES

1.1 Advantages and disadvantages of using software decoys 11
1.2 Comparative characteristics of applications built on the first level of Blockchain 22
2.1 Security services and general measures 28
2.2 Comparison of RSA and ECC key lengths in bits 29
2.3 Features of Blockchain-based e-government 29
3.1 Comparative characteristics of machine learning algorithms for anomaly analysis 45
3.2 Comparative characteristics of GPT 3.5 and GPT 4.0 for event analysis 49
3.3 Notations used for the Blockchain-based decoy system 52
3.4 Description of system actions 53
4.1 Transitional dependencies of the states of the links 61
5.1 Service response rate when there is no attack 78
5.2 Attack rate before the service stops accepting requests 79
6.1 Criterion: Information system development phase (P) 83
6.2 Criterion: Security measures (SC) 83
6.3 Criterion: Information classification (IC) 84
6.4 Notations used to model the system 90
6.5 Description of system actions 90
6.6 System components 92

LIST OF FIGURES

1.1 Schematic diagram of the trend of the use of software decoys by users 9
1.2 Classification of software decoys into groups 10
1.3 Interaction levels of a software decoy system 12
1.4 Honeypot system diagram 15
1.5 Scheme of the deception system 16
1.6 Decoy system diagram 17
1.7 Scheme of an agent system 18
1.8 Decoy, agent and bait in the system 18
1.9 Scheme of a false user system 19
2.1 Blockchain technology application sectors 24
3.1 NIDS principle 34
3.2 HIDS operation principle 35
3.3 SIEM operation principle 36
3.4 EDR operation principle 37
3.5 Schematic representation of a threat research system 38
3.6 Schematic representation of a cloud-based cybercrime investigation system 40
3.7 Model comparison 44
3.8 Scheme of a potentially vulnerable environment 47
3.9 Comparison of analysis time of GPT 3.5 and GPT 4.0 models 50
3.10 Schematic representation of the components of the cybercrime investigation system 51
3.11 Blockchain-based decoy system diagram 52
4.1 Dynamic distributed software decoy system model 55
4.2 Different main hosts: *a* – main host in $Table_0$; *b* – main host in $Table_1$ 56
4.3 Decentralized communication mechanism 58
4.4 Comparison: *a* – primary host in $Table_0$ before migration; *b* – primary host in $Table_1$ after migration 58
4.5 Method of dynamic system built using software decoys 60
4.6 Transitional states of links 61
4.7 Behavior during communication 63
5.1 Interface of the developed Blockchain controller program 68
5.2 Distribution of real services between hosts 69
5.3 Information obtained from a sniffer attack 69
5.4 First scan: result 70
5.5 Second scan: result 70
5.6 TCP throughput comparison 71

5.7 Comparison of average throughput rate 72
5.8 Response time: MySQL 73
5.9 Response time: Apache 74
5.10 Response time: VsFTPd 75
5.11 Response time: Nginx 75
5.12 Comparison of the effectiveness of the centralized model and the developed model during the attack 76
5.13 The result of attacks on the centralized model 77
5.14 The result of attacks on the developed model 78
6.1 Schematic representation of a threat research system 81
6.2 Schematic representation of the vulnerability management system 84
6.3 Cybercrime investigation system model for information systems infrastructure components 85
6.4 Flowchart of actions 88
6.5 Sequence diagram 91
6.6 Component diagram (C&C) 93
6.7 Attack simulation environment 94
6.8 Vulnerability scan results 96
6.9 Attack using automated scanning tools 98
6.10 Injection attack 99
6.11 Directory Traversal Attack 100
6.12 Attempt to violate the program logic 101
6.13 Comparison of scan attack analysis time by a comprehensive cybercrime investigation system and a traditional approach 102
6.14 Comparison of Directory Traversal attack analysis time by a comprehensive cybercrime investigation system and a traditional approach 103
6.15 Comparison of the analysis time of attacks with violation of logic by a comprehensive cybercrime investigation system and a traditional approach 104
6.16 Analysis of the effectiveness of information security event detection by the cybercrime investigation system for components of the information systems infrastructure 105
6.17 Analysis of the effectiveness of information security event detection by the cybercrime investigation system for components of the information systems infrastructure 106

CIRCLE OF READERS AND SCOPE OF APPLICATION

Methodology for Cooperative Conflict Interaction Modeling of Security System Agents is proposed. The concept of modeling the structure and functioning of the security system of critical infrastructure facilities is demonstrated for development of a model for the implementation of a terrorist act and the degree of security of the cyber system of a critical infrastructure object. Lotka-Volterra model are used for assessing the level of security of critical infrastructure facilities. The method for assessing forecast of social impact in regional communities as a case of socio-cyber-physical systems security concept is presented.

Methodological aspects of providing information security of an individual, society and state in social networking services are investigated. Identification of threats to the information security of the state in the text content of social networking services is used for information security profiles of actors in social networking services, their classification, and information-psychological influence on actors and approaches to its evaluation. The model of conflictual interaction of civic movements in social networking services.

Counteracting the strategic manipulation of public opinion in decision-making by actors of social networking services based on the conceptual model for managed self-organization in social networking services are developed.

The problems of physical access to critical infrastructure based on analysis of biometric protection systems as a class of authentication systems are introduces. Algorithms for thinning the critical infrastructure identification system and their software are implemented.

For teachers, scientific and engineering staff in the field of cybersecurity, information technology, social engineering, communication systems, computer technology, automated control systems and economic information security, as well as for adjuncts, graduate students and senior students of relevant specialties.

INTRODUCTION

The development and improvement of information protection methods to counteract unauthorized access to information facilities and their telecommunications networks is a critically important task in the conditions of modern cyberspace, which is constantly under attack from attackers. In the conditions of rapid technological development and the growth of the number of cyber threats, traditional protection methods, such as firewalls and intrusion detection systems (IDS), no longer provide a sufficient level of security. This necessitates the implementation of innovative solutions that will allow for the effective detection and counteraction of new threats.

One of the promising areas is the use of blockchain-based honeypot systems. Honeypots are a powerful tool for detecting and studying the behavior of attackers, as they imitate vulnerable systems and attract attackers. Using blockchain to implement honeypots provides an additional level of security and decentralization, which makes it difficult to detect and compromise them. Blockchain allows to store and process data about attacks in a distributed registry, which increases the reliability and stability of the system. In addition, blockchain technologies can ensure transparency and immutability of data, which is important for storing evidence of cyberattacks and further investigation.

Another important area is the study of cybercrimes using artificial intelligence (AI). AI-based systems are able to analyze large volumes of data in real time, detect anomalies and unusual activity, which indicates potential cyberattacks. Thanks to machine learning algorithms, such systems can adapt to new types of threats, providing more effective protection of information systems. The use of AI allows to automate the processes of detecting and responding to threats, reducing the need for human intervention and increasing the rate of response.

The relevance of the research lies in the need to develop new methods of information protection that take into account modern challenges and threats. The integration of blockchain and artificial intelligence technologies into the field of cybersecurity will allow the creation of comprehensive protection systems that will be able to effectively counteract unauthorized access and ensure a high level of security of information objects and their telecommunication networks. This will contribute not only to increasing the security of critical infrastructure, but also to the development of scientific approaches to ensuring cybersecurity on a global scale. Recently, there has been a growing interest in security and information protection for network systems that contain valuable data and resources that need protection from attackers. Security experts often use honeypots and honeynets to protect network systems. Honeypot is an outstanding technology for detecting new hacking methods by attackers. According to Spitzner, the founder of the Honeynet project, "a honeypot is a security resource whose value lies in being investigated, attacked, or compromised". In other words, a decoy is a mirror of an object placed on the network to lure attackers. Honeypots are typically virtual machines that mimic real machines and services with open ports that can be found on a typical system or server.

As a proactive defense mechanism, honeypot is an indispensable tool for ensuring network security in applications such as the Internet of Things (IoT), wireless sensor networks (WSN), and transport networks.

Through proactivity and control, honeypots lure attackers into interacting with fake system resources, preventing attacks on valuable resources. Compared with traditional methods such as firewalls and intrusion detection systems (IDS), honeypots have attracted widespread attention among cybersecurity forces due to their effectiveness. Honeypots are divided into two categories: static and dynamic. Static honeypots have a fixed configuration and response, which allows attackers to easily detect them. Dynamic honeypots, due to their variable configuration, are better able to deceive attackers and learn their attack modes. The problem of improving the attack prevention and intervention system of attackers has been studied by many scholars. Despite a large amount of research, there are still unresolved issues such as a centralized security management system, the computing power of security systems, and the human factor. Through data integration and analysis, a dynamic configuration scheme can be created and implemented in the local area network for the deployment of a corporate network. The result is implemented in a real system or network deployment.

The research paper used the centralized management of the Puppet system to automate the configuration of servers. A VMware virtual machine was used to implement automated honeypot solutions. The research provided targets with services such as Apache web server, MYSQL server, FTP, and SMTP. A centralized Logstash server was used to process and index logs, Elasticsearch was used to store logs, and Kibana was used to search and visualize logs.

Attackers use tools such as SubSeven, Nmap, and LoftCrack to scan, identify, and penetrate corporate systems. Firewalls are installed to prevent unauthorized access, but they cannot prevent attacks from the Internet. An intrusion detection system (IDS) scans network traffic and identifies exploits and vulnerabilities, is able to display warnings, log events, and inform administrators about possible attacks. On the other hand, an intrusion prevention system tries to prevent known and some unknown attacks. However, an IDS can generate thousands of warnings every day, some of which are false positives. This makes it difficult to detect real threats and protect assets, so human intervention is required. Building networks based on Blockchain is a complex task that requires significant knowledge and experience. Using ready-made solutions, such as forking from Ethereum or Solana, is not suitable for implementing information protection systems. It is necessary to develop your own first-level Blockchain along with security policies, authorization protocols, and user authentication. In today's information security environment, creating your own first-level Blockchain becomes a necessity for several reasons. First, off-the-shelf solutions, such as Ethereum or Solana forks, often do not meet the specific security requirements needed to protect information in corporate systems or critical infrastructure. They usually have generic protocols that are difficult to adapt for highly specialized tasks.

Creating your own blockchain allows to implement specialized security mechanisms, including:

– security policies that meet the unique needs of an organization or industry;

– authorization and authentication protocols designed to address modern threats;

– decentralization system that reduces the risk of a single point of failure.

In addition, your own Blockchain allows to integrate encryption algorithms designed specifically to protect confidential data and ensures the transparency and immutability of information in the system. This contributes to a more effective investigation of cyberattacks, as each operation is recorded in a distributed registry.

It is important to emphasize that in the field of cybersecurity, saving on the development of such systems often becomes the cause of successful attacks. As statistics show, 80 % of successful penetrations occur precisely due to insufficient funding for security systems. Thus, creating your own blockchain is an investment that allows not only to protect data, but also to strengthen the overall cyber resilience of the organization.

There should be no savings in the concept of security of important objects, because 80 % of successful attacks occur due to savings in protection. The remaining 20 % includes the human factor.

The paper considers all the advantages and disadvantages of protecting cyber systems built on Blockchain technology and develops a conceptual solution using Blockchain-based software decoys with the property of host decentralization. The cybercrime and cybersecurity landscape between 2020 and 2023 demonstrate the rapid development of threats. According to Qualys research, 25 % of vulnerabilities were used on the day of their publication, and 75 % – within 19 days. Varonis statistics show that 94 % of malware is delivered via email, and hackers attack an average of 26,000 times a day. Remote work adds complexity, increasing the time to detect and eliminate violations.

Integrating advanced technologies such as AI is key to effectively combating cybercrime. AI algorithms can process large amounts of data, detect anomalies, automate threat responses, and provide real-time security event information. This allows for the detection of unusual activities such as network traffic irregularities, suspicious emails, or financial transactions. The use of AI in information security highlights the importance of machine learning in threat detection. By training on cyber threat datasets, AI models can effectively detect malicious software in real-time.

The problem of improving cybercrime investigation systems and using AI has been studied by many scholars. Despite significant research, there are a number of unresolved issues, including ensuring compliance with international standards and the impact of the DevSecOps approach on cybercrime investigation. The experiments conducted comparing anomaly detection models provide an opportunity to create a prototype of a cybercrime investigation system that effectively detects suspicious activity and notifies information security analysts for detailed analysis.

This paper addresses the problem of cybercrime analysis, in particular the use of AI models to reduce the duration of analysis without reducing the effectiveness of cyberattack detection. Challenges include the integration of different systems, such as threat research and vulnerability management, as well as ensuring compliance with international standards. The advantages and disadvantages of cybercrime investigation using AI models and the DevSecOps approach are described.

1 ANALYSIS OF THE PROBLEM OF USING SOFTWARE DECOYS AND REVIEW OF EXISTING SOLUTIONS FOR INFORMATION ACTIVITY OBJECTS

1.1 ANALYSIS OF THE CURRENT STATE OF RESEARCH AND PUBLICATIONS

Over the past few years, many different uses for software decoys have been proposed. Some are used to waste hackers' time, others to reduce spam activity or to deceive attackers, and some to analyze the steps of a hacker's intrusion. For several years, the security community has used honeypots to analyze the various methods used by attackers.

L. Spitzner introduced two types of honeypots: low-level interaction adventures, which are computer software that emulates operating systems and services that are typically used in a production environment in an organization, and high-level adventures, which involve deploying real operating systems on real or virtual machines, which are also used in security research [1]. L. Spitzner also identified two critical requirements for a honeynet architecture:

1) data controls reduce risk by ensuring that once an attacker infiltrates the honeynet systems, these compromised systems cannot be used to attack or harm other systems;

2) data capture ensures that security experts can detect and capture all actions performed by an attacker, even if they are encrypted [1].

This study used high-interaction manipulators to collect data from real systems.

B. Sobesto et al. presented DarkNOC, a management and monitoring tool for complex honeynets. It consists of various types of Honeypots, as well as other data collection devices. They designed a solution that can handle the large amount of malicious traffic received by a large Honeynet network and effectively provide a user-friendly web interface to display potential compromised hosts to network security administrators, and also provides an overall security status of the network. In this study, Kibana implemented a user-friendly web interface to visually display attacks with high frequency. D. Dittrich noted that a distributed network of decoys can grow in size. A single honeypot cannot effectively monitor and control attacks [2–4].

Analyzing attack data originating from a large number of individual honeypots in a network is challenging. D. Dittrich recommended using Manuka, a front-end and back-end database tool, to address this complex problem [5]. The database contains system attributes that complement the search for operating system type, version, and installed services. He stated that Manuka can take just one person to install honeypots, load them into a database, and rapidly deploy them across a distributed network. Similarly, this study used the open-source automation management tool Puppet to deploy servers and services to honeypots without manual intervention [6–8].

Weiler proposed a system that large organizations could use to protect against distributed denial-of-service attacks. His research relied on honeypot technology, which has two advantages.

Weiler includes a demilitarized zone network that implements services such as web, mail, ftp, and DNS for access via external networks. The organization's local area network (LAN) is in another zone protected

by a firewall. The infrastructure then introduced a new system: a honeypot, which would mimic the internal network and attract DDoS attackers. Weiler also created internal systems within the organization to act as a decoy. "For example, if an attacker's compromised packets are detected on a corporate web server, the packets are sent to the honeypot for processing. The response that the attacker receives is indistinguishable from the real response from the web server" [8].

The above design solved three problems: attacks can be detected, attack packets can be actively directed to the honeypot, and the honeypot can mimic the organization's network infrastructure, at least parts known to the attacker. J. Wilson, D. Maimon, B. Sobesto, and T. Zucker studied the effect of a surveillance banner on an attacked computer system, which reduced the likelihood of command entry during longer initial intrusion incidents. The study also found that the likelihood of the number of commands entered during subsequent intrusion incidents on the same system was determined by the presence of the surveillance banner and whether the commands were entered as a result of previous intrusion incidents [9].

J. Wilson et al.'s results show that the average number of intrusion incidents per computer is 4.44 and 4.48 for computers without banners and surveillance banners, respectively. This is not a statistically significant difference; however, the presence did affect the severity of intrusion incidents. The study also found that attackers who received a surveillance banner and spent at least 50 seconds on the system were 8 % less likely to enter commands into the system than their counterparts who did not receive a surveillance banner. Of those who had not previously entered commands into the system, 38 % of those with a surveillance banner and 47 % of those who did not enter commands entered commands during a second intrusion incident. Furthermore, of those who entered commands into the system, 67 % of those with a surveillance banner and 63 % of those without surveillance banners entered commands during a second intrusion incident [9].

The study comes from a randomized controlled trial conducted at a large public university in the United States of America. Over a 7-month experimental period, 660 target computer systems were deployed after being compromised by system intruders. These systems caused 2,942 intrusion incidents. Again, they said that each of these systems was randomly assigned to display a video surveillance banner (n = 324) or not display surveillance (n = 336) upon each login to the respective system. The video surveillance banner is shown on the left, exactly as the attackers saw it, with the message: "This system is under continuous surveillance. All user activity is monitored and recorded".

R. Stockman, J. Rein, and M. Hayle used the open-source Honeynet system to study the impact of a system banner message on hackers. The study was conducted over a 25-day period, collecting data on nearly 200,000 events [10].

According to R. Stockman et al., 510 logins were allowed through password tampering, and of the 510 logins, 280 received a warning banner and 230 received no banner at all. The average duration of the attempts for those with the warning banner was 15.29 seconds and for those without was 23.45 seconds. The average duration for those with the warning banner was 9 seconds and for those without was 11 seconds. In addition, the attackers did nothing during the login because the system only recorded a few commands. However, they stated that the study did not allow attackers to log in as root, which may have led to a decrease in activity [10].

The work by R. Stockman et al. used manual intervention to configure the honeynet network. However, this study investigated ways to create targets that are of greater interest to hackers to attack services such as web servers or database servers, which appear to be resources of real value to attackers [10].

H. Döring proposed five test cases in which a honeypot could be deployed in different ways in a particular environment, and explained their individual attributes. According to his study, "the number of attacks that occur in a secure environment is smaller than the number of attacks from an unsecured environment, at least they should. Therefore, after comparing the results, one should focus on the environment" [11].

The results of D. Döring's study provided a practical approach to honeypot deployment. He discussed four phases: analysis, design, implementation, and conclusion. According to him, the first phase of development begins with gathering knowledge about Honeypots and defining requirements [11].

J. Hoke and M. Bikas presented an intrusion detection system (IDS) that uses a genetic algorithm (GA) to effectively detect different types of network attacks. Their approach used evolutionary theory to filter traffic data and thus reduce complexity [20]. They used a randomly generated population of chromosomes that represent all possible solutions to a problem, which are considered as candidates. From each chromosome, different positions are encoded as bits, symbols, or numbers [12].

It is also worth noting that some honeypot schemes are dynamic. The dynamic property is mainly reflected in the configuration and deployment [21]. M. Kuwatly proposes a dynamic honeypot scheme and uses Nmap, POf, and Snort for active detection and passive fingerprinting [2]. Honeyd and some highly interactive honeypots are used to simulate the network and redirect network traffic, respectively. The dynamic honeypot mechanism interacts with the above-mentioned modules, dynamically configuring Honeyd and providing a configurable interface. Hassan et al. dynamically configures honeypots to simulate a real industrial network in real time (i.e., the honeypot is a decoy of the real system) and allocates unused IP addresses to the Honeyd cluster [4].

H. Saeedi et al. studies dynamic decoy management. According to the data collected from routers, firewalls, IDS and honeypot, the honeypot configuration is dynamically adjusted to adapt to the network environment. W. Fan et al. combines a high-interactive honeypot with a low-interactive one. The adaptive honeynet scheme is implemented by modeling some operating systems. The key module of this scheme is the Honeybrid gateway, which contains the decision-making and forwarding parts. The former is used to capture and forward certain network traffic to Honeyd. The latter aims to redirect the Honeyd flow to a high-interactive software decoy. The above works discuss the dynamic honeypot configuration scheme [4]. There are some works on the dynamic honeypot deployment. C. Hecker and B. Hay propose a honeypot deployment automation scheme. Active and passive network flow detection technologies are used to monitor the network. User configuration information is stored in a database, which can serve as a classification criterion for creating a new honeynet, limiting the throughput and the target IP range of the network [13]. A dynamic Honeypot scheme based on machine learning, Honeyvers, was proposed by M. Fraunholz et al. The network environment is scanned and the equipment is classified to determine the exact number of honeypots, thus automatically generating configuration information and subsequent deployment of honeypots. To solve the problem caused by uneven honeynet deployment, Fan et al. put forward a multi-virtual network management architecture that generates specific honeynet information based on different queries. An individual honeynet is automatically deployed by a set of tools [7, 14].

These dynamic honeypot schemes pretend to fit into the network environment self-adaptively and focus on deceiving attackers. However, the location of these decoys seems to be fixed after determining the configuration or deployment information. With the development of anti-honeypot technology, all these designs are likely to be found and accounted for. Due to the location transformation property in the proposed scheme of this paper, these dynamic honeypot configurations are different from others. In the proposed scheme, even if attackers discover the honeypot, they cannot find the real nodes and users.

C. White and B. Mazanec's book "Understanding Cyber Warfare: Politics, Policy, and Strategy" describes the evolution of cyber conflict: from its roots of covert espionage to the sophisticated digital offensives of today. It analyzes how states and non-state actors use cyber potential to achieve their goals. "Understanding Cyber Warfare: Politics, Policy, and Strategy" recognizes that cyber conflict is not a one-sided affair. The book identifies the actors involved: from nation-states with powerful digital arsenals to shadowy non-state groups and even private companies with their own strategic goals. Understanding their motives and capabilities is essential for analyzing cyber conflicts and predicting future threats [15].

Due to the relevance of the topic of cyber threats, there has been a huge increase in cybersecurity research to support cyber programs and avoid the major security threats that these programs face. In the paper "Cybersecurity threats and vulnerabilities: A systematic study" the authors conducted a systematic review of research articles published between 2014 and 2019. They used a search strategy to identify relevant articles and then analyzed them based on a predefined set of criteria. This allowed them to identify the most common critical cybersecurity threats and vulnerabilities, as well as the relationships between them. The study identified a wide range of cybersecurity threats and vulnerabilities. In particular, these are threats that target network infrastructure, such as denial-of-service attacks, malware, and botnets; threats targeting software applications, such as SQL injection attacks, XSS attacks, and buffer overflows; threats targeting data, such as data breaches, unauthorized access, and data loss; threats targeting the operating system and hardware, such as zero-day attacks, embedded software vulnerabilities, and physical security breaches. The results of this study are of great importance to cybersecurity professionals. By understanding the most common critical threats and vulnerabilities, organizations can develop more effective defense strategies [16].

An important aspect of countering cyberattacks, once they are understood, is a well-defined defense strategy. The article "On the most common threats to cyber systems" by H. Kettani and P. Wainwright offers an analysis that includes general trends in the complexity of attacks, actors, and the maturity of organizations' skills and capabilities to defend against attacks. The authors highlight the concept of increasingly sophisticated attacks orchestrated by state actors, while traditional defenses often fail to keep up. H. Kettani and P. Wainwright emphasize the need for automation and proactive threat analysis to stay ahead of these sophisticated adversaries, urging organizations to implement state-of-the-art security measures and invest in qualified personnel [17].

The ever-increasing level of sophisticated, high-velocity cyberattacks presents new challenges to those working in cybersecurity. In the ever-evolving field of software development, security can no longer be simply a cosmetic addition to the finished product. A. Koskinen, in her article "DevSecOps: Building security into the foundation of DevOps" explores the core DevOps principles of culture, automation, measurement, and sharing (CAMS). The author interprets and applies these principles through security controls. Case

studies are also presented, demonstrating the successful implementation of DevSecOps in various organizations [18]. These examples demonstrate DevSecOps as an approach that embeds Static Application Security Analysis (SASA), Dynamic Application Security Analysis (DASA), and Software Composition Analysis (SCA) into the development framework, which can lead to not only more secure but also more efficient and resilient software. DevSecOps can be defined as a cultural approach to improving and accelerating the achievement of business value through effective collaboration between development, security, and operations teams. The article "Self-Service Cybersecurity Monitoring as an Enabler for DevSecOps" is dedicated to self-service cybersecurity monitoring as a tool for implementing security practices in a DevOps environment. This study proves that a cybersecurity monitoring infrastructure using the DevSecOps approach allowed the detection of threats such as denial-of-service attacks and helped to better predict spoofing issues. This infrastructure is implemented according to DevOps best practices: it is automated using scripts and configuration files, and its deployment is automated using virtualization and containerization technology [18].

The attempt to protect data by implementing best practices such as DevSecOps and information security technologies has led to a growing role for cybersecurity. Due to the ever-growing threat landscape, risks, and challenges, in recent years, research by scholars has focused on the role of Artificial Intelligence and its impact on cybersecurity at a large scale. The article "Machine learning for intelligent data analysis and automation in cybersecurity: Current and future prospects" describes how machine learning can be used to automate cybersecurity tasks and improve data analysis for threat detection. This study provides an opportunity to determine that machine learning is revolutionizing cybersecurity by automating tasks and providing a deeper understanding of security data. R. Sahini, in his work "AI and critical infrastructure security: Challenges and opportunities", explores the use of artificial intelligence to protect critical infrastructure, such as power grids and transportation systems, from cyberattacks. This study confirms that artificial intelligence can be a valuable tool for protecting critical infrastructure, but it also creates new vulnerabilities that need to be addressed [19]. The relevance of using machine learning is also highlighted in the article "Artificial intelligence and international security: Opportunities, risks, and the arms race" by M. Horowitz and J. Michael. However, the authors analyze the impact of artificial intelligence (AI) on international security, including potential benefits and risks, and indicate that artificial intelligence has the potential to improve both offensive and defensive capabilities in warfare, which raises concerns about a new arms race [20].

Research data indicate that AI can have both positive and negative impacts on organizational security, in particular, S. Zhou et al. in the article "Adversarial attacks and defenses in deep learning: From a perspective of cybersecurity" explores how attackers can manipulate deep learning models (e.g., facial recognition) to their advantage, and how to defend against these attacks. In the course of this study, the author concludes that adversarial attacks pose a serious threat to the security of deep learning, but various defense mechanisms are being developed [21]. Despite the fact that AI can improve the state of cybersecurity, its nature requires caution. As the analyzed materials emphasize, innovative approaches can increase task automation, data analysis, and even help prevent cyberattacks. However, the need for a comprehensive approach to cybercrime research can help with international security and critical infrastructure issues, given the potential for arms races and exploitation by attackers. Ultimately, both the DevSecOps approach and AI models act as transformative forces in cybersecurity, requiring responsible development, deployment, and further research to reap the benefits and mitigate the risks.

1.2 OVERVIEW OF SOFTWARE DECOYS AND THEIR PURPOSES

First of all, a honeypot is a computer system. It has files, directories, just like a real computer. However, the purpose of the computer is to lure hackers into it to observe and monitor their behavior. Therefore, it is possible to define it as a fake system that looks like a real system. They are different from other security systems because they do not only find a single solution to a specific problem, but also have the power to apply a variety of security problems and find multiple approaches to them. For example, they can be used to record malicious actions in a compromised system, they can also be used to study new threats to users and generate ideas on how to get rid of these problems. Honeypots can be divided according to their goals and level of interaction. If to look at the goals of the decoys, it is possible to see that there are two types of decoys: research and production [22–46].

Growth trends of decoy technology users:

2010–2012: moderate growth, as Honeypot/Deception technologies are still in their early stages of development and are more actively used by security researchers and organizations with a high level of cybersecurity knowledge.

2013–2016: rapid growth, as a result of wider recognition of Honeypot/Deception technologies and their effectiveness in combating cybercrime. The development of open source and commercial products also contributes to the spread of these technologies.

2017–2023: steady growth with an increased rate, as more companies and organizations integrate Honeypot/Deception technologies into their cybersecurity systems. Cybersecurity professionals and companies are increasingly using these solutions to proactively detect and block threats.

A schematic diagram of the growth trend can be seen in **Fig. 1.1**.

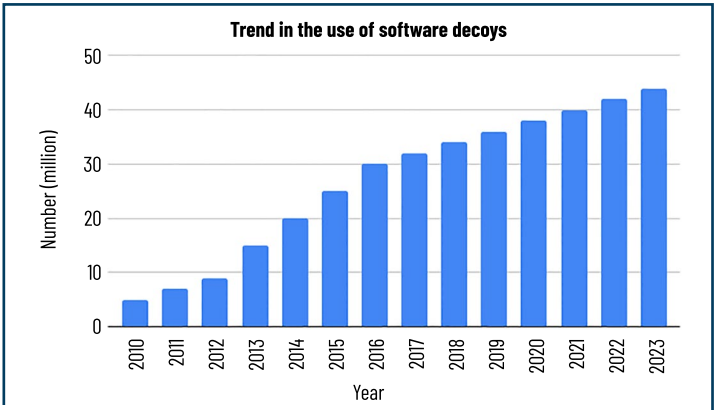


Fig. 1.1 Schematic diagram of the trend of the use of software decoys by users

Research decoys are mainly used by military, research and government organizations. They collect a huge amount of information. Their goal is to identify new threats and learn more about the motives and

techniques of Blackhat. The goal is to learn how to better protect the system, they do not bring any direct value to the security of the organization [23].

Production honeypots are used to protect the company from attacks, they are implemented inside the production network to improve overall security. They collect a limited amount of information, mainly low-interaction decoys are used. Thus, the security administrator closely monitors the movements of the hacker and tries to reduce the risks that may arise from him/her for the company. At this stage, it is possible to try to discuss and clarify the risks of using production software decoys. Since during the security testing of systems existing in the organization, unexpected actions may occur, for example, unauthorized use of other systems using honeypot functions. If the network administrator is not aware of this problem, they create big problems for the organization [24].

The groups of software decoys can be divided into: prevention, detection and response, which are schematically depicted in **Fig. 1.2.**

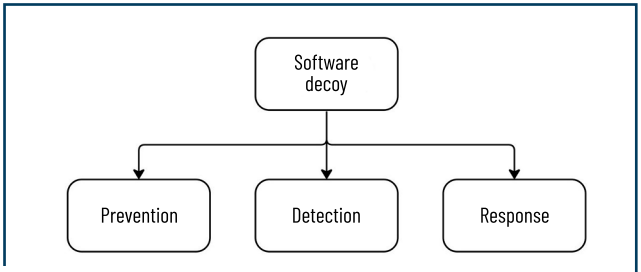


Fig. 1.2 Classification of software decoys into groups

Prevention is the first thing to consider in our security model. By definition, it means preventing hackers from breaking into the system. Therefore, it is possible to prevent them from gaining access to the system. There are many ways to do this securely. It is possible to use a firewall to manage network traffic and set some rules to block or allow it. Using authentication methods, digital certificates, or having strong passwords are the most common and well-known methods of protection. There are also encryption algorithms that encrypt data [25]. This is a good way to use it because it encrypts messages and makes them unreadable. The relationship between the use of prevention and software decoys can be explained as follows. If a hacker understands that the company he/she is trying to hack uses honeypots, and he/she is aware of today's security issues, this will make them think about it. For a hacker, this will be confusing and scary. Even if a company uses the methods discussed in the first paragraph to stay safe, it is still good to have a honeypot in the organization because security issues are concerned and handled professionally. Since security is very important, it is always good to be aware. There is no tolerance when there is a problem, it can cause great damage to any company. Since every company has confidential and important data, it is necessary to protect the data from attackers [26].

Detection is the act of detecting any malicious activity in the system. It is assumed that the prevention did not work somehow, the system was hacked by a hacker. There are several ways to detect these attacks.

A well-known solution for detection is Network Intrusion Detection Systems. This technology will help users know if the network is hacked, but it will not prevent hackers from attacking the system. For companies, such detection systems are expensive. At this stage, manipulators are valuable for monitoring the activity [2].

Response. At this stage, we are sure that an attack has occurred and there will be a response to it. This is where the investigation begins. When a hacker compromises a system, he/she leaves traces. With the right tools, it is possible to process the data in such a way that we have some clues about what happened to the system. It is possible to look at the log files and try to investigate what happened [5].

1.2.1 ANALYSIS OF ADVANTAGES AND DISADVANTAGES OF SOFTWARE DECOYS

There are many security solutions available on the market. Everyone can browse the various options on the Internet and find the most suitable solution for their needs. The advantages and disadvantages are shown in **Table 1.1**.

Table 1.1 Advantages and disadvantages of using software decoys

Advantages	Disadvantages
Honeypots can capture attacks and provide information about the type of attack, and if necessary, it is possible to see additional information about the attack through logs. By reviewing them, it is possible to see new attacks and create new security solutions. Additional checks can be obtained by looking at the type of malicious behavior. This helps to understand more attacks that may occur. Honeypots are not cumbersome in terms of data collection	Data capture is only possible when the hacker is actively attacking the system. If he/she is not attacking the system, it is impossible to capture information. If another system is being attacked, the honeypot will not be able to identify it. Therefore, attacks not on the honeypot system can damage other systems and cause big problems
They only deal with incoming malicious traffic. Therefore, the information that is captured is not as much as all traffic. Focusing only on malicious traffic makes the investigation much easier. Therefore, this makes decoys very useful. There is no need for a huge data store for a single malicious traffic	There is a drawback to fingerprinting in honeypots. It is easy for an experienced hacker to understand whether he/she is attacking the honeypot system or the real system. Fingerprinting allows to distinguish between the two. This is not a desired outcome
There is no need to maintain new technologies. Any computer can be used as a honeypot system. Therefore, creating such a system does not require additional costs. They are easy to understand, configure and install. They do not have complex algorithms. No need to update or change things	Honeypot can be used as a zombie to reach other systems and compromise them. This can be very dangerous
Since honeypots can capture anything malicious, they can also capture new tools for detecting attacks. This gives more ideas and depth to the topic, proving that different perspectives can be discovered and applied to security solutions	Honeypots are passive defenses. They do not stop the attack in real time and may be ineffective in detecting attacks that are not directed at them. In addition, attackers can use honeypots to test new types of attacks

1.2.2 INTERACTION LEVELS IN SOFTWARE DECOYS

Since honeypots have been classified according to their purposes, it is now time to take a closer look at the interaction levels. The interaction level refers to the extent to which a hacker can interact with the system [26]. The more data one wants to collect, the more interaction level is required. The more interaction level also brings more risks to the network security. There are three categories of interaction levels in honeypots. They are called low interaction, medium interaction, and high interaction, which are shown in **Fig. 1.3.**

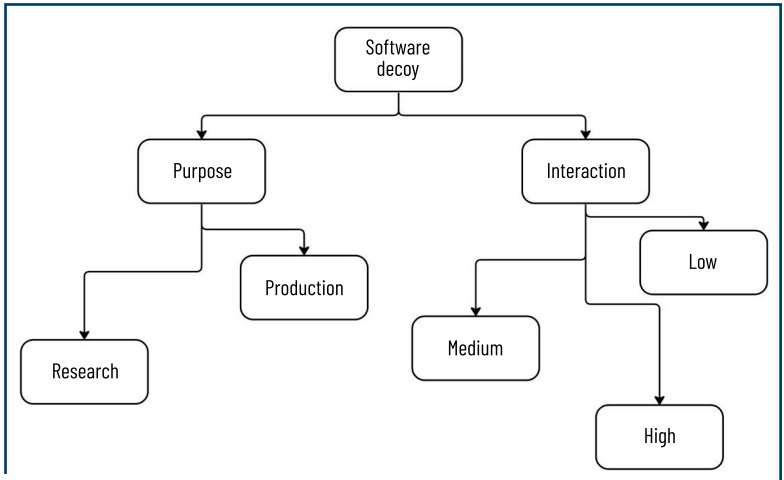


Fig. 1.3 Interaction levels of a software decoy system

Low-interaction decoys can obtain the least amount of data compared to other honeypot systems. They are limited, so the risk to the attacker is also not proportional. First of all, there is no operating system to deal with. They can be used to detect new worms or viruses and analyze the traffic passing through the network. Low-interaction devices are easy to configure and understand.

Medium-interaction decoys are more advanced than low-interaction decoys. But this time, more information and more sophisticated attacks can be obtained from the hacker. Since it is more advanced, it has more ways to enter the security of the system so that the hacker can access it. Mwcollect, honeytrap, and Nepenthes are some of the medium-interaction decoys in use today [27].

High-interaction honeypots are the most advanced honeypots. Unlike low and medium interaction honeypots, there is an operating system. As a result, the hacker can do anything. Proportionally more data can be obtained from the hacker's activities. However, it is the riskiest when it comes to security because it gives the hacker such access that he/she has no restrictions. Such honeypots are time-consuming and difficult to maintain. Honeywall is a good example of a high interaction honeypot [28].

1.2.3 OBSTACLES TO THE DEPLOYMENT OF SOFTWARE DECOYS

HPT is rapidly evolving and is beginning to be used in large organizations that are required to protect highly sensitive or easily replaceable assets such as money. Businesses and governments are increasing their surveillance of the workplace, both physically and digitally. It is interesting to articulate whether the barriers to deployment are technological or organizational? From a technological perspective, most honeypot deployments are still very experimental, even in commercially supported solutions [29]. HPT is likely to be more robust in some respects than the intrusion prevention system (IPS) packages offered by commercial vendors that some sites use. The adoption of IPS may be a result of its being marketed as a silver bullet replacement for the inability of IDS to detect new forms of attacks [30]. IPS, IDS and firewall technology provide a convenient and secure way to fend off external hackers at the perimeter of the network interface.

In contrast, honeypots are almost the antithesis of IDS, IPS, and firewall technologies, which take an approach that follows the principle celebrated by Sun Tzu: keep your enemies more closely watched than your friends, by bringing them inside a defensive perimeter and monitoring their every move. This mode of operation represents a significant shift in thinking from the perimeter mentality that underlies most modern network security. In the same way that modern warfare has evolved into network effects and operations, network security requires a similar paradigm shift from regulated compartments to network-centric tactics. One of the most common responses to the deployment of honeypots is the legal concept of "capture" [31].

This area is highly contentious and, of course, varies greatly depending on the jurisdiction in which the concept of capture is being considered. Internal honeypot systems do indeed pose various ethical and moral issues for an organization, but no more so than existing network monitoring schemes. For example, most organizations now record users' email and World Wide Web usage, often without the user's knowledge. The capture argument can be applied to these existing workplace surveillance scenarios. However, it is fair to say that properly constructed honeypot systems, placed on a corporate network that is tightly controlled by appropriate policies and procedures, should not pose any issues in this regard [32]. The honeypot in this situation should only encounter users who are directly violating company policy and whose actions are intentional, malicious, and deliberate. This would include policies that restrict users to accessing the services for which they are intended and that they are legitimately using. In addition, the honeypot system itself would contain warnings via banners or pop-up messages that access to these systems is intended for authorized personnel only and that actions outside of this point would violate company policy. If the user then decides to explore or attempts to compromise the system, they have made a conscious decision to do so, and this cannot be called hijacking [33].

Unlike external honeypots, internally deployed honeypots can capture existing behavior and threat patterns within an organization's network. Internal honeypots may not have to respond to new types of malicious code or new exploits that target an external honeypot due to the depth of their network within the organization [34]. Consequently, the level of expertise required to effectively manage one may be lower, as they can leverage existing defaults in systems that mitigate known threats. Honeyfiles are a method that is not tied to any specific exploit and is aimed directly at determining whether a file has been accessed. The internal flaw of IDSs is that they look for a specific binary sequence or rule set compromise for a single

instance of behavior, to which they then typically respond with a single action, such as dropping a connection that either succeeds or fails. IDSs generally have poor mechanisms for allowing forensic reconstruction of an incident beyond the scope of the IDS's actions, as their primary function is to protect the system, not to collect data. However, HPTs are designed to track and monitor any behavior without harm, even if there is interaction, for example, with the IDS in a honeypot system. The HPT should be designed to capture the actions of attackers and the impact on the system not only at the network level, but also at the system and application levels. This approach allows security professionals to reconstruct the sequence of events or actions that led to the incident, as all the data should be sufficient for the entire forensic reconstruction of the incident. This should allow internal security personnel to adopt a learning paradigm for remediating security incidents by analyzing this data.

Internal users can also compromise valuable internal systems that are not visible or accessible to the Internet through conventional countermeasures, unable to detect this abuse because they are typically focused on the external point of egress to the Internet. Many organizations use firewall and virtual local area network (VLAN) systems to control traffic flows to these high-value networks, and some even deploy IDS [18]. IDS and firewall technologies can suffer from a lack of forensic completeness due to the way they operate. There are often elements of trust at the organizational level that can make deploying and even managing these advanced countermeasures unpleasant [35]. This problem is perhaps exacerbated by the perception of attacks as external. Deploying internal decoy software in these valuable network segments can significantly reduce the risk profiles of the organization. This is because HPT can detect behavior before it becomes a problem and act as an early warning system for security administrators. For example, HPT can also effectively record unknown or unspecified security flaws in existing systems, such as incorrect rights management settings for a particular server or service, allowing an internal user to gain evidentiary capabilities.

1.3 THE PROBLEM OF USING DECOYS AND DECOYS TO PROTECT DATA IN COMPUTER NETWORKS

Honeypot can be considered the first embodiment of Deception technology, and they appeared in the late eighties and early nineties. Honeypot is a network object whose sole purpose is to attract an attacker and be attacked. When a Honeypot is attacked, it registers this and stores all the actions of the attacker. In the future, this data helps to analyze the attacker's path. The second goal of Honeypot is to delay the attacker's progress through the network, forcing it to spend time studying the fake resource. Let's present a diagram of the Honeypot system in **Fig. 1.4** [36].

Honeypot can be a full-fledged operating system that emulates an employee's workplace or a server, or a separate service. Understanding the abilities of attackers is important for building a protection system that can detect them.

Here are some ways in which attackers detect the presence of Honeypots [37–40]:

– if access to the system seems too easy, it may be fake;

– usually, systems connected to the Internet do not have unnecessary ports and services, any deviation from this configuration may indicate a trap;

– if the system still has default settings, this increases the likelihood of using a Honeypot;

– if there is a lot of free space on the hard drive or very little software, this may be a decoy;

– if the folder names are trivial (for example, "Salaries", "Customer Data", "Passwords"), it is obvious that the systems are aimed at luring attackers.

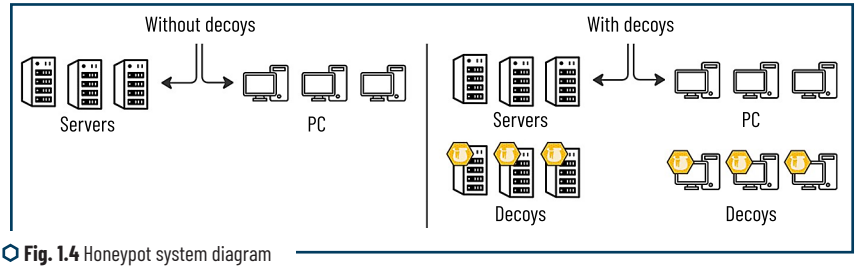


Fig. 1.4 Honeypot system diagram

All these signals tell attackers that the system may be fake. These systems also have a number of disadvantages:

– each fake server must be configured separately;

– Honeypots do not interact with each other and with elements of the real infrastructure. They leave no trace and are difficult for a hacker to detect;

– Honeypots are usually not integrated into a centralized system.

This technology has gradually been replaced by another, more advanced and intelligent one – Deception.

Social engineering and phishing attacks are examples of how any class of solutions, including decoys, can be circumvented.

Many modern attacks begin by delivering a "decoy" to the user, such as a phishing email that they open on their computer. This allows the malware to penetrate the internal network and allows the attacker to move on to planning and executing the next stage of the attack.

Honeypots are not able to handle a phishing attack in the same way that users do. Therefore, Honeypots will not be able to provoke and detect an attack using such a vector. Unlike Honeypots, next-generation deception technologies can automatically change the decoy environment, not leaving it static, as befits a real network, in which user and network data naturally change. At the same time, deception technologies detect an attacker in just three to four steps in the network, even if deception elements are not deployed on every node [41].

Next-generation deception technologies provide users with powerful real-time attack detection and forensic collection functionality, with virtually no false positives, and attackers never know they are being monitored. Decoys are also effective for detecting attackers, but they have a much lower detection rate for real threats, generate much more false positives, and do not provide forensics from the real nodes that attackers use to attack.

Deception refers to solutions of the Intrusion Detection System (IDS) class. The main purpose of such a system is to detect unwanted attempts to access the network. In other words, Deception helps to detect network attacks. Honeypot is a separate network resource that does not interact with anyone, but only waits

for an attacker to record its actions [12]. On the other hand, Deception technology is a centralized system for managing fake network objects, which are usually called traps (decoys). Each trap is, in fact, a separate decoy, but they are all connected to a central server. The scheme of Deception technology is shown in **Fig. 1.5**.

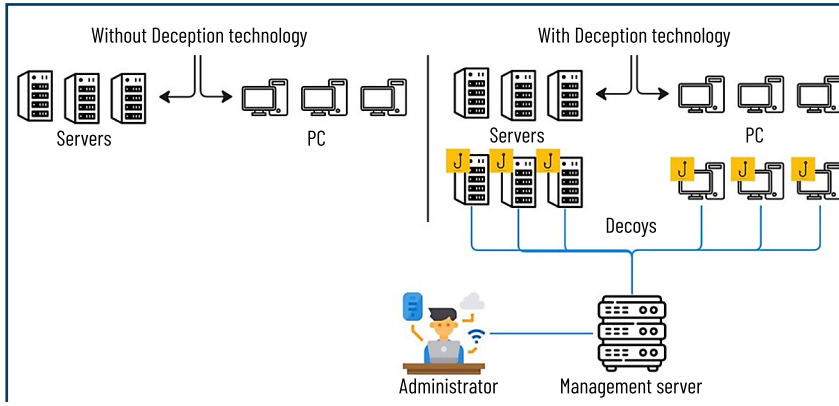


Fig. 1.5 Scheme of the deception system

Such solutions usually have a convenient interface for managing traps. The operator can create traps with the desired set of emulated network services, in the selected subnet, with the desired method of obtaining an IP address, etc. [42]. Traps and the services emulated on them maintain constant communication with the server. Like Honeypots, traps in Deception do not provide legitimate interaction with the network (except for interaction with other Deception components). The trap will notify the server of any attempts to interact with it: this serves as an indicator of an attack. In this case, the operator can instantly receive a notification about the event [43]. It will indicate the details of what happened: the address and port of the source and target, the protocol of interaction, the response time, and so on. Additional modules in Deception technology can also provide manual or automated incident response capabilities (**Fig. 1.6**).

The concept of deception can include other things. Some components help simplify configuration and deployment automation, others make the decoys look more like real network services, and still others draw hackers' attention to fake targets. Some components can perform related tasks, such as responding to incidents, collecting indicators of compromise from workstations, and scanning them for vulnerable software [44].

An agent is a program that is installed on real workstations or servers of users. It is able to communicate with the deception server, execute its commands, or transmit user data to a control center. Among the solutions of the Deception class are both products that contain an agent and those that do not (**Fig. 1.7**).

Tasks for agents may include:

– collection of data on the state of the workstation;

– distribution of decoys;

– emulation of network activity;

– incident response (manual or automated);
– collection of data for forensics;
– other – as needed by the customer and the developer's imagination.

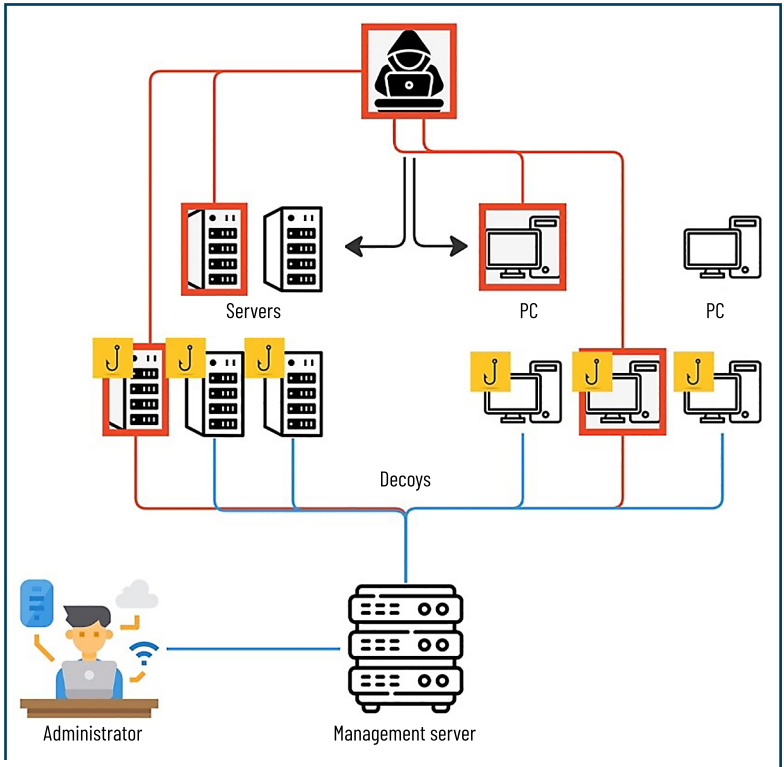


Fig. 1.6 Decoy system diagram

The activities of agents should be hidden from the person working at the computer [43]. First, the user can intentionally or accidentally remove the agent or its components. Second, the presence of unknown (or to some extent known – if the user is warned about it) software on the workstation can cause a feeling of discomfort. Third, everything that the user sees will be seen by an attacker who has gained access to this computer.

Agent decisions within the framework of deception should be made in such a way that the user does not see either the agent or traces of his/her vital activity (or at least tries to minimize it). Therefore, agents usually run in a privileged mode, like a driver for Windows or a kernel module for Linux. This allows, for example, to intercept system calls to ensure stealth, and also does not allow the user to remove the agent or prevent it from running [44].

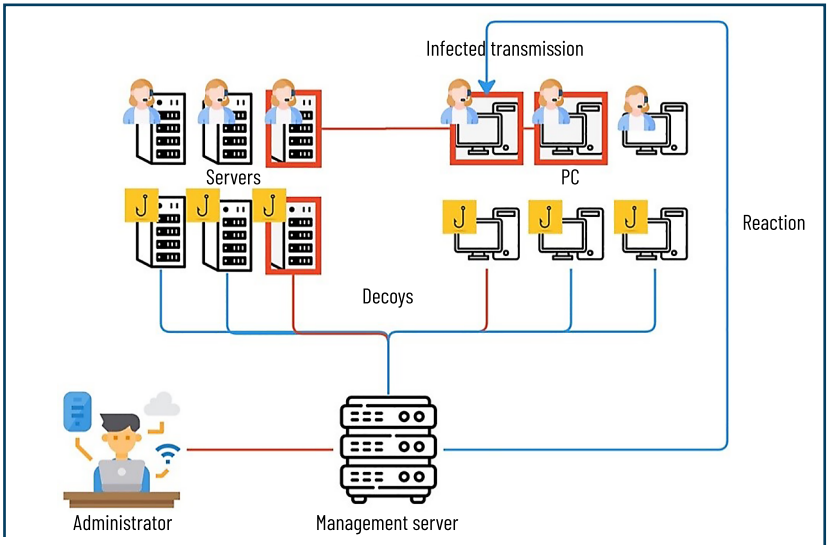


Fig. 1.7 Scheme of an agent system

Honeypot contains links and data to access a fake network resource. An attacker, having found such a link and authorization data, of course, wants to check what kind of service it is. It falls into the trap, and then an event signal is triggered (**Fig. 1.8**).

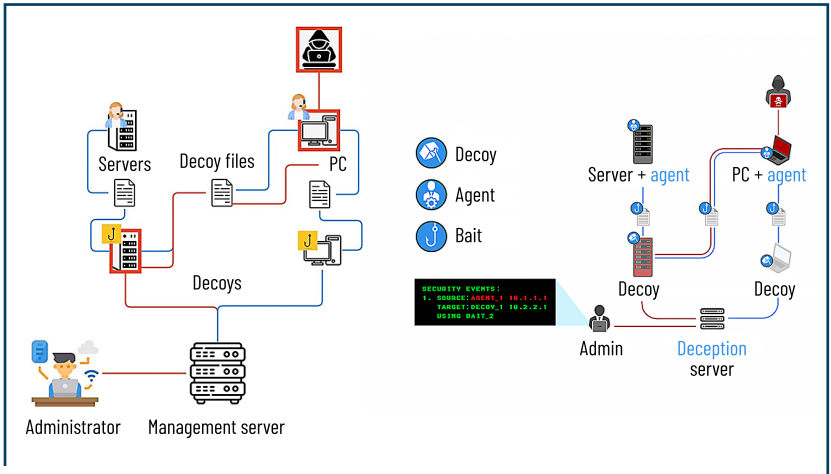
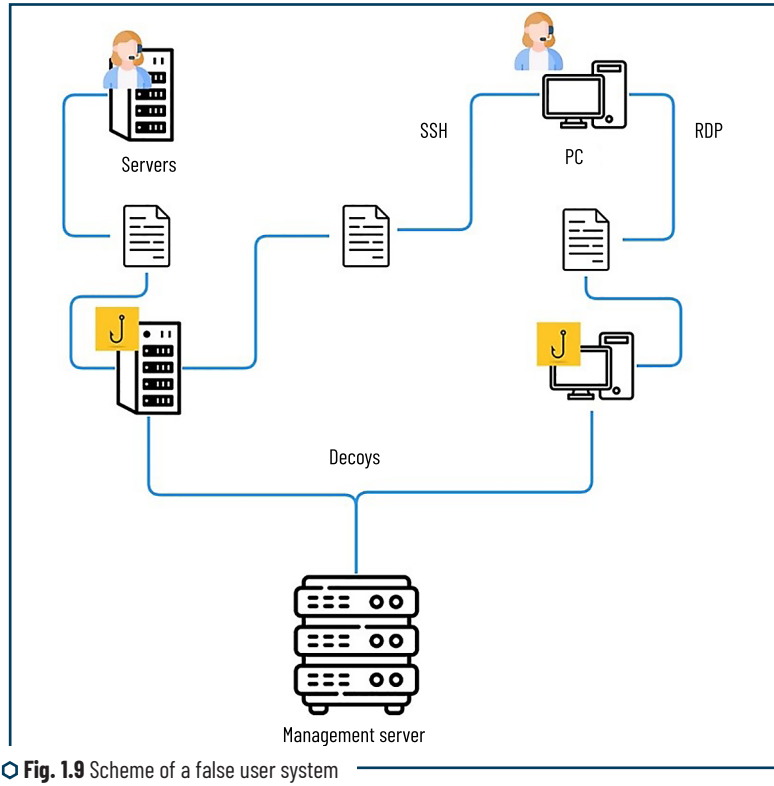


Fig. 1.8 Decoy, agent and bait in the system

The types and methods of placing the decoy depend on the type of trap to which the decoy leads. Decoys can be distributed in several ways. If agents are present in the deception, they are tasked with scattering the decoys [45]. In this case, the process can be easily automated: the control server sends a command to the agent, and the latter performs the necessary actions to install the decoy.

Some decoys may require an email address, domain address or something else. The problem can be solved by maintaining a database of fake network users. There are different approaches to maintaining such a database (**Fig. 1.9**).



⬡ **Fig. 1.9** Scheme of a false user system

For example, Deception can be integrated with a traffic analysis system. This allows to recognize the presence of authorization data in network traffic, find common features in them and generate users similar to real ones, according to identified rules.

NFC technology can be used together with software decoys or deception technologies, such as Honeypots, to increase the level of security of computer networks, especially in the context of authentication and authorization of individuals, for example:

1. Increasing the level of authentication security: the use of NFC tags as a component of dual authentication (password + tag) provides an additional layer of protection for decentralized hosts and computer networks, which reduces the likelihood of unauthorized access to the system.

2. Protection against phishing and other attacks: using NFC tags in combination with passwords can protect users from phishing, brute force attacks and other methods aimed at stealing passwords. Even if an attacker gains access to the password, without physical access to the NFC tag, he/she will not be able to log in to the system.

3. Distracting attackers: using NFC tags in software decoys and Honeypots can encourage attackers to spend their time and resources on analyzing and attacking decoys, instead of trying to gain access to real systems. This will help ensure the security of real corporate assets.

4. Collecting information about attackers: using NFC technologies, it is possible to create a decoy that stores information about attackers and their actions. This will allow to collect data about the activity of attackers, providing knowledge about threats and the ability to take preventive measures to protect computer networks. Analyzing the data obtained can help identify weaknesses in the system and ensure their timely elimination.

5. Contactless communication: NFC technology works on the basis of contactless communication between devices over short distances, which provides rate and ease of use. This can be used to create decoys that are easy to deploy and control, while providing reliable protection against unauthorized access.

6. Protection against remote interception: since NFC has a short range, typically less than 10 centimeters, it is difficult for attackers to remotely intercept the signal. This means that attackers need to have physical access to the decoy or tag to carry out an attack, which increases the level of security of the system.

In summary, the use of NFC technology together with software decoys and deception technologies such as Honeypots can increase the level of security of computer networks and decentralized hosts. Double authentication with a password and an NFC tag, as well as distracting attackers, collecting information about them, and reliable protection against remote attacks make NFC technology a valuable tool for improving the security of information systems [46].

1.4 ANALYSIS OF THE USE OF BLOCKCHAIN TECHNOLOGY

Blockchain is the underlying technology that powers many cryptocurrencies, such as Bitcoin and Ethereum, but its unique way of securely recording and transmitting information has broader applications beyond cryptocurrencies [48].

Blockchain is a type of distributed ledger. Distributed ledger technology (DLT) allows records to be maintained on multiple computers, known as "nodes". Any Blockchain user can be a node, but it requires a lot of computing power to operate. Nodes verify, approve, and store data in a ledger. This is different from traditional recordkeeping methods, which store data in a central location, such as a computer server.

Blockchain organizes information added to a ledger into blocks, or groups of data.

Each block can only contain a certain amount of information, so new blocks are constantly added to the ledger, forming a chain [49].

Each block has its own unique identifier, a cryptographic "hash". A hash not only protects the information in a block from anyone without the necessary code, but it also protects the block's place along the chain by identifying the block that came before it.

A cryptographic hash is "a set of numbers and letters that can be up to 64 digits long", says V. Agarwal, a partner in PwC's financial services advisory practice. "It's a unique code that allows the pieces of the puzzle to fit together" [50].

Once information is added to the Blockchain and encrypted with a hash, it becomes permanent and immutable. Each node has its own record of the complete timeline of data along the Blockchain, going back to the beginning. If someone tampers with or hacks one computer and manipulates the data for their own benefit, it won't change the information stored on other nodes. The altered record can be easily distinguished and corrected because it doesn't match the majority.

The way the system works, it is virtually impossible for someone to replicate the computational power that goes on behind the scenes to reverse engineer it and somehow figure out what the hashes are [51].

Here is an example of how the Blockchain is used to verify and record Bitcoin transactions:

1. Consumer purchases Bitcoin.

2. Transaction data is sent through a decentralized network of Bitcoin nodes.

3. Nodes confirm the transaction.

4. Once approved, the transaction is grouped with other transactions to form a block, which is added to the ever-growing chain of transactions.

5. The completed block is encrypted, and the transaction record is permanent; it cannot be deleted or altered on the Blockchain.

The Bitcoin Blockchain is public, meaning that anyone who owns a Bitcoin can view the transaction record. While it may be difficult to trace the identity of an account, the record shows which accounts are making transactions on the Blockchain. Public Blockchains also allow any user with the necessary computing power to participate in validating and recording transactions in the Blockchain as a node [50].

But not all Blockchains are public. Blockchains can be designed as private systems, so the owner can limit who can make changes or additions to the Blockchain. While the pool of participants may be smaller in a private Blockchain, it is still decentralized among those who participate. Private Blockchains ensure the security of any data stored in the database using the same encryption techniques.

The idea of a secure, decentralized permanent record of information has sparked interest in a number of industries and potentially holds solutions to many of the security, recordkeeping, and data ownership issues we face today. Blockchain gives the technology to securely move information, says V. Agarwal, and have almost complete confidence in the authenticity of any information you want to protect. Let's consider, for example, the stories that have been circulating in recent weeks about memes and celebrities who have made money off of digital property by selling NFTs (non-fungible tokens). Because the underlying Blockchain record is immutable, NFTs allow sellers to prove the authenticity of a digital asset. When you buy an NFT, that transaction is added to the Blockchain ledger and becomes a verified record of ownership. For those

who want to be able to verify the authenticity of a digital work, Blockchain helps to value digital art and collectibles in the same way as their physical counterparts. In theory, this leads to creators retaining value through the objects that receive royalties for copies of digital art [51–60].

Such uses illustrate the appeal of Blockchain not only for security, but also for information integrity. Blockchain has the potential to give people more security and confidence in this. Companies and governments around the world continue to test and implement Blockchain technology, but it won't happen overnight. If we ever reach a point where a government currency is based on Blockchain or medical records are converted to Blockchain, it won't be anytime soon.

1.4.1 USING BLOCKCHAIN SYSTEMS AS SECURITY IN REAL–WORLD EXAMPLES

Bitcoin also works on Blockchain technology. The entire Blockchain is stored on a huge network of computers. So, no one person has control over the history. Let's say a user learns cryptocurrency-related data from Crypto Head [52], buys Bitcoins, and pays others using Bitcoins. What happens next? The computers on the Bitcoin Blockchain rush to verify the accuracy of the transaction. This step certifies everything that happened on the chain. This way, no one can go back and change things. Therefore, the Blockchain cannot be easily forged.

This complex structure provides Blockchain technology with the potential to be the most secure form of information storage and exchange on the Internet that we have discovered. This is why innovators have started to apply the technology in various sectors to prevent fraud and increase data security.

Guardtime removes the need for keys for verification. Instead, they distribute every piece of data across nodes throughout the system. If someone tries to change the data, the system analyzes the entire mass of chains, compares them with the metadata package, and then excludes all that do not match.

This means that the only way to erase the entire Blockchain is to destroy each individual node. If only one node continues to work with the correct data, the entire system can be restored, even if all other nodes are compromised. In the context of hybrid warfare, this allows to protect any confidential information from destruction or distortion.

The Guardtime system works in such a way that it can always determine when changes have been made to the data, and constantly checks the changes. This ensures that there is no discrete way to interfere with the blocks in the chain, and the data remains uncompromised.

When choosing ready-made solutions or when creating your own first-level Blockchain, it is necessary to understand how they are formed and what they consist of. The **Table 1.2** compares several of the most popular first-level Blockchain systems.

Table 1.2 Comparative characteristics of applications built on the first level of Blockchain

Features	Bitcoin	Etherum	IOTA
1	**2**	**3**	**4**
Fully released	Yes	Yes	Yes

Continuation of Table 1.2

1	2	3	4
Mining permissions	Public	Public, private, hybrid	Public
Secure transaction system	Yes	Yes	There are trusted validator nodes
External app support	Financial only	Yes	Yes
Data privacy	No	No	Yes
Forking capability	Yes	Yes	No
No fees	No	No	Yes
Key management	No	No	No
Personal ID management	No	No	No
User authentication	Digital signature	Digital signature	Digital signature

2 STUDYING THE USE OF BLOCKCHAIN PROPERTIES TO PREVENT UNAUTHORIZED ACCESS TO INFORMATION ACTIVITY FACILITIES

2.1 STUDYING THE POSSIBILITIES OF USING BLOCKCHAIN TO PROTECT DATA IN INFRASTRUCTURE FACILITIES

Blockchain has emerged as one of the most secure forms of transactions in the digital network space. As designed and conceived, the technology has been recognized for ensuring the integrity of information. If used correctly, many sectors can benefit from it, as depicted in **Fig. 2.1**.

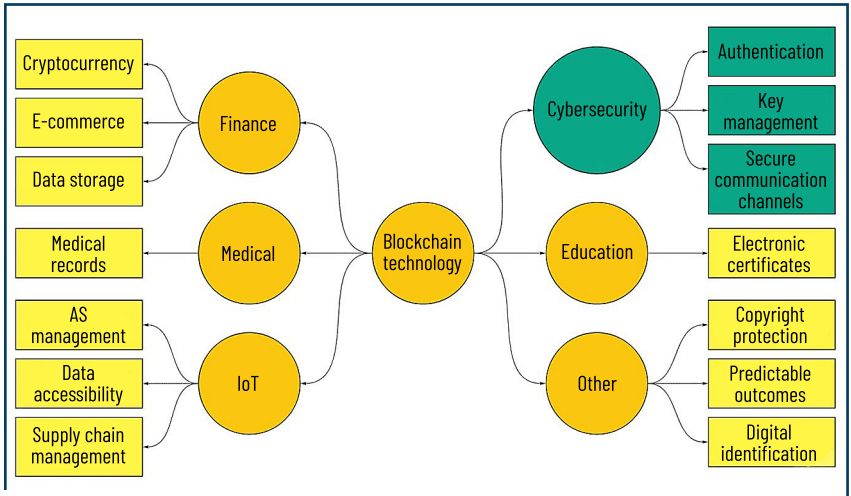


Fig. 2.1 Blockchain technology application sectors

With the potential to be practical for many applications, Blockchain can be implemented in many cases. One of the best uses would be to use its integrity guarantee to create cybersecurity solutions for many other technologies. The following are some of the future beneficial use cases of Blockchain to enhance cybersecurity [56–83].

Protecting private messages. As the Internet turns the world into a global city, more people are joining social networks. The number of social networks is also growing. More social applications are launched with each dawn as conversational commerce gains popularity. During these interactions, a huge amount of metadata is collected. Most users of social networking platforms protect their services and their data with weak, insecure passwords.

Most messaging companies are turning to Blockchain to protect user data as a better option than the end-to-end encryption they currently use. Blockchain can be used to create a standard security

protocol. To enable inter-messaging capabilities, Blockchain can be used to form a unified API framework [42, 59, 69–76].

Recently, there have been numerous attacks on social platforms such as Twitter and Facebook. These attacks resulted in data breaches, millions of accounts were compromised, and user information fell into the wrong hands. Blockchain technologies, if well implemented in these messaging systems, can prevent such future cyberattacks, especially when it comes to internal structures and networks of the state, where this is a critical factor [57].

IoT security. Hackers are increasingly using peripheral devices such as thermostats and routers to gain access to shared systems. With the current obsession with artificial intelligence (AI), it has become easier for hackers to gain access to shared systems such as home automation through edge devices such as "smart" switches. In most cases, a large number of these IoT devices have sketchy security features.

In this case, Blockchain can be used to secure such shared systems or devices by decentralizing their administration. The approach would empower the device to make its own security decisions. Independence from a central administrator or authority makes peripheral devices more secure by detecting and acting on suspicious commands from unknown networks.

Hackers typically break into the central administration of a device and automatically gain full control over the devices and systems. By decentralizing such device authority systems, Blockchain ensures that such attacks are harder to execute (if even possible) [58–60].

DNS and DDoS Protection. A Distributed Denial of Service (DDoS) attack occurs when users of a targeted resource, such as a network resource, server, or website, are denied access to or service to the targeted resource. These attacks shut down or slow down the resource systems. On the other hand, an intact Domain Name System (DNS) is highly centralized, making it an ideal target for hackers who can hack into the connection between an IP address and a website name. This attack makes the website inaccessible and even creates redirects to other fraudulent websites. Blockchain can be used to mitigate such attacks by decentralizing DNS records. By implementing decentralized solutions, Blockchain would remove the single points of vulnerability that hackers exploit.

Decentralization of Storage. Data breaches and theft are becoming a major apparent cause of concern for organizations. Most companies still use a centralized form of storage. To gain access to all the data stored in these systems, a hacker simply uses a single vulnerability. This attack leaves the attacker with sensitive data, such as a business's financial records. Using Blockchain, sensitive data can be protected by providing a decentralized form of data storage. This method will make it more difficult, if not impossible, for hackers to penetrate data storage systems. Many companies that use data storage services agree that Blockchain can protect data from hackers. Apollo Currency Team is a good example of an organization that has already adopted Blockchain technology in its systems (the Apollo Data Cloud).

Software. Blockchain can be used to ensure the integrity of software downloads to prevent third-party intrusion. Just as MD5 hashes are used, Blockchain can be used to verify activities such as firmware updates, installers, and patches to prevent malware from entering computers. In the MD5 scenario, the new software ID is compared to hashes available on the vendor's websites. This method is not completely reliable, as the hashes available on the vendor's platform may already be compromised. However, in the case of

Blockchain technology, the hashes are permanently recorded on the Blockchain. The information recorded on the technology is immutable, and therefore, Blockchain can be more effective in verifying the integrity of the software by comparing it to the hashes on the Blockchain.

Cyber-physical infrastructure verification. Data tampering, system misconfiguration, and component failures have compromised the integrity of information received from cyber-physical systems. However, the integrity and verification capabilities of Blockchain technology can be used to authenticate the status of any cyber-physical infrastructure. Information received about infrastructure components through Blockchain can be more reliable for the entire chain of custody.

Data transmission protection. Blockchain can be used in the future to prevent unauthorized access to data during transmission. By using the full encryption function of this technology, data transmission can be protected to prevent access by attackers, be it an individual or an organization. This approach will lead to an overall increase in the confidence and integrity of data transmitted through Blockchain. Hackers with malicious intent maintain data during transmission to modify it or completely remove its existence. This leaves a huge gap in inefficient communication channels such as email.

All of the above-mentioned areas of application of Blockchain technology can be implemented as additional protection of critical infrastructure facilities of Ukraine, in dual-purpose networks, special information networks, as well as for the protection of confidential data when transmitting them over open or dual-purpose networks [73]. It can be argued that in conditions of hybrid warfare or during electronic warfare, when the ability to transmit military, intelligence data, operational data, etc. is not possible over closed networks, then Blockchain technology can be used. MaidSafe is a similar company based in the UK. Their goal is also to decentralize the Internet and create something like an alternative Internet, where users will be able to run programs, store data and do everything else that they usually do online, but in a more secure environment. When registering with this service, users can choose how much of their personal data storage space they want to allocate to the network. The system then provides safecoin, a cryptocurrency, to compensate users for the value (space) they offer to the network. Every file hosted on the MaidSafe network is encrypted, fragmented, and distributed among users. The only person who can make the data readable again is its owner, ensuring that no one other than the authorized owner can access the data.

2.2 THE RELATIONSHIP BETWEEN BLOCKCHAIN TECHNOLOGY AND AI, BIG DATA AND OTHER ADVANCED TECHNOLOGIES

The development and application of Blockchain technology would not be possible without the support of next-generation information technology infrastructures such as AI, big data, cloud computing, and the Internet of Things [79]. In turn, Blockchain technology has also contributed to the development of these information technologies. AI and Blockchain are expected to complement each other with their respective advantages:

1. AI can help solve the problems faced by Blockchain in terms of autonomy, efficiency, energy efficiency, and intelligence, especially the reliability of data in AI applications, so that AI can focus more on algorithms. In addition, artificial intelligence can more effectively manage the autonomous organization of

Blockchain, expand and improve the functions and efficiency of smart contracts, and optimize Blockchain operations to improve security, efficiency, and energy efficiency.

2. Blockchain can provide distributed artificial intelligence, realize mutual invocation of various artificial intelligence functions, accelerate the development of artificial intelligence, break the currently closed development mode, and promote data sharing. In addition, Blockchain can also be used for audit log data models, which provides more reliable predictions, etc.

3. When the number of nodes reaches a certain scale, the attack cost will also be huge, which will make it difficult to implement the attack. In this sense, the problem of data leakage will be effectively solved.

4. The operation and maintenance cost of IoT is greatly reduced. With Blockchain technology, IoT can transmit data in a point-to-point manner.

5. Direct connection without the use of central processors. Distributed computing can be used to process hundreds of millions of transactions. Moreover, the computing power, storage capacity, and throughput of hundreds of millions of idle equipment will greatly reduce the cost of computing and storage.

6. No third-party verification is required. Blockchain technology can help solve the problems of scalability, single point of failure, timestamping, recording, confidentiality, trust and reliability in a completely consistent manner. In a fully decentralized trusted digital infrastructure, IoT equipment can operate independently without the need to obtain any centralized authorization.

7. IoT security is guaranteed. Blockchain can record all actions of terminal devices. Since the recorded information can never be overwritten, data security and user privacy will be put under effective protection.

8. Cloud computing services are characterized by large scale, high reliability, low cost, flexibility and on-demand delivery. Combined with the decentralization and data protection of Blockchain, it has the potential to promote the widespread application of Blockchain technology. In the future, based on Infrastructure and Services (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS), Blockchain as a Service (BaaS) will be established to integrate Blockchain technology into cloud computing platforms and create a BaaS cloud services market, thus providing stable and reliable cloud computing platforms for decentralized applications.

2.3 SECURITY AND PRIVACY ANALYSIS

Every information system must guarantee confidentiality, integrity and availability of services. Confidentiality is achieved when information is not disclosed to unauthorized users; Integrity is achieved by protecting information from any form of modification, while availability means that information is available when needed and is free from DoS or DDoS or other similar service disruptions. This section presents a theoretical qualitative analysis of the security and confidentiality performance of a Blockchain-based e-government system [75–80].

Records stored in the proposed system are protected by public-key cryptography, which protects against malicious attempts to modify and/or unauthorized access, while network users are assigned private keys to verify and sign transactions. Encryption and digital signature are used in the network to ensure security,

confidentiality and access control of stored records. Furthermore, most Blockchain consensus algorithms (in this case DPoS) require an attacker to control at least 51 % of the peer-to-peer network to change a record, which is usually impossible to achieve. More precisely, to change any block in the blockchain, the attacker must change every copy of that block in the network and then convince the majority of nodes that the new block is valid. Furthermore, to increase the confidentiality of the data stored in the proposed network, all user blocks are hashed and the unreadable transaction hashes are stored in the Blockchain [81–83].

The proposed system is a decentralized P2P system where user data is stored in different nodes, which guarantees the availability of the system, avoiding any single point of failure. Using DPoS, it is difficult for any attacker to launch DDoS or DoS attacks against the system, since registration is required for a node to start exchanging information with the rest of the peer-to-peer network. Any transactions received from a network node are verified by witnesses, making it difficult for malicious nodes to initiate malicious connections [84–86].

The security services and corresponding general measures provided by the proposed framework are summarized in **Table 2.1**, which ensures adequate confidentiality and security of transactions. To improve computational efficiency, user devices will run light clients to store transactions rather than a full copy of the Blockchain, which is expensive in terms of storage. E-government devices are expected to be computationally powerful with sufficient capacity to store and efficiently process user records. The network is able to offer the performance provided by Blockchain technology and the DPoS consensus protocol, such as scalability, rate, compatibility, and transparency, and it can handle a large number of transactions.

Table 2.1 Security services and general measures

Security Service	Countermeasures
Authentication	Blockchain address and digital signature
Access control	Digital signature and encryption
Confidentiality	Encryption
Integrity	Encryption and digital signature
Non-repudiation	Encryption and digital signature
Availability	Distributed/decentralized
Trust	Decentralization, encryption and digital signature

The Elliptic Curve Cryptography (ECC) approach is used to implement encryption and digital signature in the proposed framework, which is a common practice for most existing Blockchain technologies such as Bitcoin and Ethereum. Let's note that ECC and RSA (Rivest-Shamir-Adleman) offer a similar level of security, but ECC consumes a much smaller number of bits. For example, a 256-bit key in ECC offers the same level of security as RSA, which uses a 3072-bit key. A shorter key usually means low CPU consumption, low memory usage, and fast key generation. These advantages also contribute to fast transaction creation and block sealing. A summary of the key length research between RSA and ECC is given in **Table 2.2**. 256-bit ECC keys are widely used in Blockchain technology because they can provide the required level of security for most applications.

Table 2.2 Comparison of RSA and ECC key lengths in bits

RSA key length	ECC key length	Approximate ratio (RSA:ECC)
1024	160	6:1
2048	224	9:1
3072	256	12:1
7680	348	20:1
15360	512	30:1

In addition to security and privacy, Blockchain-based e-government also provides a number of other benefits, summarized in **Table 2.3**. These features make Blockchain technology a promising direction in the implementation of e-government, which can provide a convenient, secure and fault-tolerant communication channel between the public sector and citizens. The indirect benefits brought by Blockchain technologies, such as reducing bureaucracy, eliminating paper, reducing transaction costs and controlling corruption, can transform the governance ecosystem with a higher degree of trust from citizens [81].

Table 2.3 Features of Blockchain-based e-government

Feature	Explanation
Reduced human error	Devices and user credentials are authenticated in advance before accessing the network
Increased public trust	People have direct control over their information and all network participants are authenticated
Improved scalability	The system can be easily scaled as it allows new devices and users to be automatically added to the network according to a consensus mechanism
Increased reliability	Data is stored across multiple servers/locations. Consensus protocol ensures that data can only be changed with the consent of all participants
Increased fault tolerance	Avoids single point of failure and the system is therefore resistant to malware, DoS and DDoS attacks
Improved auditability	Easy to track the history of all transactions as they remain immutable in the network
Improved auditability	All new transactions are verified by all network participants before being added to the Blockchain
Information ownership	Individuals are responsible for providing access to their information
Improved access to information	Information is stored in multiple locations, which improves easy and fast access
Improved data quality	All transactions and records stored in the system are verified in advance, which makes the stored information authentic with the required quality
Increased transparency	All nodes in the network share the same copy of the Blockchain and new transactions are added based on a consensus mechanism
Reduced operating costs	No third party is required to process transactions
Improved efficiency and rate	Anyone in the network can access all records subject to the availability privilege and new records are propagated to all participating nodes

2.4 ADVANTAGES AND DISADVANTAGES OF USING BLOCKCHAIN TECHNOLOGY TO PROTECT CYBER SYSTEMS AND TELECOMMUNICATIONS NETWORKS

The technological complexity of building a Blockchain to protect cyber systems raises some concerns about implementation, security, and resilience. Let's take a closer look at the pros and cons of Blockchain in the context of cybersecurity and data protection.

The main advantages of Blockchain technology for cybersecurity are as follows:

1. Decentralization – due to the peer-to-peer network, there is no need for third-party verification, as any user can see network transactions.

2. Traceability – all transactions in the Blockchain are digitally signed and time-stamped, so network users can easily track transaction history and track accounts at any historical point in time. This feature also allows a company to have reliable information about assets or product distribution.

3. Privacy – privacy of network participants is high thanks to public-key cryptography, which authenticates users and encrypts their transactions. However, some Blockchain-based startups are going even further and improving this technology. For example, Guardtime has developed a Keyless Signature Infrastructure (KSI), which allows users to verify the validity of their signature without revealing their keys.

4. Right to be forgotten – data privacy is important, even if your information is immutable. Since there is no way to erase unnecessary information, Blockchain technology ensures the privacy of your data when you forget your key, as no one can decrypt it.

5. Fraud protection – in the event of a hack, malicious behavior is easy to detect thanks to peer-to-peer connection and distributed consensus. Blockchains are currently considered "unbreakable" because attackers can only affect the network by gaining control of 51 % of the network nodes.

6. Resilience – Blockchain technology has no single point of failure, which means that even in the event of a DDoS attack, the system will continue to operate normally thanks to multiple copies of the chain.

7. Integrity – distributed ledger protects data from modification or destruction. In addition, the technology ensures the authenticity and irreversibility of transactions. Encrypted blocks contain immutable data that is resistant to hacking.

8. Resilience – peer-to-peer nature of the technology ensures that the network will operate 24/7, even if some nodes are offline or under attack. In the event of an attack, the company can make certain nodes redundant and operate as usual.

9. Data quality – Blockchain technology cannot improve the quality of your data, but it can guarantee the accuracy and quality of data once it is encrypted in the Blockchain.

10. Secure network access – employees may need constant access to the Blockchain from multiple devices, so the company risks losing control of its private keys. To avoid risks associated with key loss or human error, Blockchain REMME provides each user and each device with a special Secure Sockets Layer certificate, which eliminates the need for passwords. This approach makes it impossible for unauthorized access to the network.

11. Secure communication – business correspondence contains sensitive data that can be effectively protected if to use Blockchain for cybersecurity. There are many startups that encrypt business

communication. For example, Obsidian uses Blockchain-based networks to mitigate vulnerabilities in end-to-end encryption. A distributed chain for messages reduces the risk of surveillance.

12. Smart contracts - programs that are based on a registry. These programs ensure that the terms of the contract are fulfilled and verify the parties. Blockchain technology can significantly increase the security standards for smart contracts as it minimizes the risks of cyberattacks and errors.

13. Availability — there is no need to store your sensitive data in one place, as Blockchain technology allows to have multiple copies of your data that are always available to network users.

14. Increase customer trust — customers will trust you more if you can provide a high level of data security. Additionally, Blockchain technology allows to instantly provide your customers with information about products and services.

Although Blockchain is changing cybersecurity, there are still some drawbacks that need to be considered:

1. Irreversibility — there is a risk that encrypted data cannot be recovered if the user loses or forgets the private key needed to decrypt it.

2. Storage (space) limitations — each block can contain no more than 1 MB of data, and Blockchain can only process 7 transactions per second. Several possible technological solutions to improve these parameters are:

— switching to alternative consensus algorithms, such as Proof of Stake (PoS), which reduce transaction processing time;

— using scalability layers, such as the Lightning Network, which allows microtransactions to be processed outside the main blockchain;

— sharding, which divides the blockchain into segments (shards) that run in parallel, reducing the load on the network;

— implementing the Segregated Witness (SegWit) protocol, which allows signature data to be stored separately from the main transaction, increasing the usable capacity of the block;

— using external storage systems such as IPFS (InterPlanetary File System) to store large data, and only the hash of this data is recorded in the blockchain;

— reducing the frequency of recording non-essential transactions, leaving the main blockchain for critical operations.

3. Cyberattack risk — although this technology significantly reduces the risk of malicious interference, it is still not a panacea for all cyber threats. If attackers manage to capture a large part of your network, you can lose your entire database.

4. Adaptation problems — although Blockchain technology can be applied to almost any business, companies may face difficulties in integrating it. Applying this technology in supply chain systems, for example, is quite difficult, as it can take a long time to recreate supply chains in the form of Blockchain and improve them. Blockchain applications may also require a complete replacement of existing systems, so companies should consider this before implementing Blockchain technology.

5. High operational costs — Blockchain technology requires significant computing power to run, which can lead to high marginal costs compared to existing systems.

6. Blockchain literacy — there is still a shortage of developers with Blockchain experience and deep knowledge of cryptography.

Overall, Blockchain technology is a breakthrough in cybersecurity because it can provide the highest level of confidentiality, availability, and data security. However, the complexity of the technology can cause difficulties in development and use in the real world. Blockchain technology relies on the latest cryptographic advances, as well as comprehensive network management experience [82].

3 RESEARCH ON THE USE OF THE PROPERTIES OF ARTIFICIAL INTELLIGENCE MODELS FOR DETECTION OF CYBERATTACKS AND ANALYSIS OF CYBERCRIMES ON INFORMATION ACTIVITY OBJECTS

3.1 ISSUES OF MODERN SOLUTIONS FOR THE STUDY OF CYBERCRIMES OF INFORMATION SYSTEMS

This section discusses the current approaches to investigating cybercrime at different levels of the information systems infrastructure and the differences in protection systems depending on the type of infrastructure.

In cloud environments, customers are responsible for access control, encryption, monitoring, and authentication mechanisms to protect their data and applications. At the same time, cloud service providers ensure the security of the underlying infrastructure. In the case of on-premises infrastructure, information security controls include physical access controls, antivirus programs, firewalls, and vulnerability management. These controls are necessary to protect against malware, data leaks, and insider threats [83–87].

However, security systems without cyberattack detection mechanisms cannot provide reliable protection for cloud, on-premises, or hybrid infrastructure. Modern standards, such as NIST, introduce the concept of "cloud forensics" – the use of scientific principles and techniques to reconstruct past events in cloud computing by identifying, collecting, storing, analyzing, and reporting digital data [88–92].

For the analysis of information security threats and cybercrime, classic solutions such as NIDS, HIDS, SIEM, EDR systems, and cloud event investigation systems are used [93–97].

The difference in cloud infrastructure monitoring is due to the prevalence of API-driven interactions, dynamic resource scaling, and specific indicators of cloud services. Instead of focusing exclusively on network connections, it is important to monitor the API calls that underlie cloud services. Therefore, a cloud solution monitoring service is added to the usual systems, which in most cases is provided by the cloud service provider. In particular, the most popular services include Azure Log Analytics and AWS GuardDuty. However, classic cybercrime investigation solutions can also be used for cloud infrastructure. For the specific features of each solution, their shortcomings and advantages, an overview of their functionality and the issues of cybercrime detection and analysis solutions is proposed [98–102].

IDS are software or hardware systems that monitor and analyze a host system or network for cyberattacks [18]. IDS technology notifies security teams about the fact of an attack, therefore it plays an important role in protecting the security of cloud computing. The same can be said about the importance of IDS systems for on-premises infrastructure, as it is the basic technology for intrusion detection. NIDS (Network Intrusion Detection System) and HIDS (Host Intrusion Detection System) are two important components of the information system infrastructure. They work together to detect and prevent various types of intrusions and threats.

NIDS is a network intrusion detection system that specializes in monitoring network traffic to detect potential intrusions and threats. The system operates at the network layer, collecting and analyzing data packets to detect suspicious events, anomalies, or known attack signatures [103–107]. The working principle of NIDS is schematically presented in **Fig. 3.1**.

The diagram in **Fig. 3.1** shows the following classes:

1. The IDS class represents the intrusion detection system itself and includes the attributes of identifier, name, description, status, and versioning. These attributes describe the IDS.

2. The NetworkTraffic class represents the network traffic data that the IDS analyzes, including attributes such as source IP, destination IP, protocol, and payload.

3. The Alert class represents security alerts generated by the IDS when suspicious activity is detected. It contains attributes such as identifier, timestamp, source, severity, and description.

4. The IDS class analyzes network traffic, meaning it analyzes network traffic data. The IDS class generates alerts, indicating that it is generating security alerts.

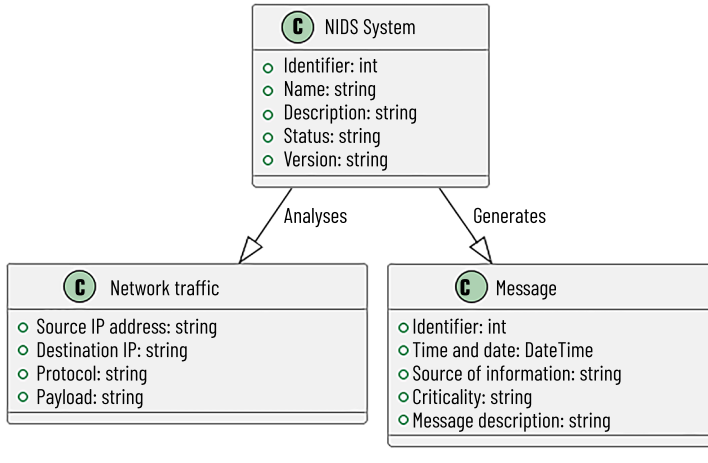


Fig. 3.1 NIDS principle

HIDS is an endpoint intrusion detection system that specializes in detecting intrusions on host systems. They operate at the host level, constantly monitoring system activities and configurations. The principle of operation of HIDS is schematically presented in **Fig. 3.2**.

The HIDS class represents the HIDS itself and includes attributes such as identifier, name, description, status, and version that describe the HIDS. The Host class represents the host system that the HIDS monitors and includes attributes such as host name, IP address, and operating system to describe the host [95–105]. The Event class represents the log data generated by the host system, including attributes such as id, time and date, source, event type, and its description. The Message class represents security alerts generated by the HIDS when it detects suspicious activity [83, 88]. It contains attributes such as identifier, timestamp, source, severity, and description. The HIDS class monitors the Host, meaning it monitors the host system. The HIDS class analyzes the log, indicating that it analyzes log data from the host. The HIDS class generates alerts, indicating that it generates security alerts based on the analysis [86–99].

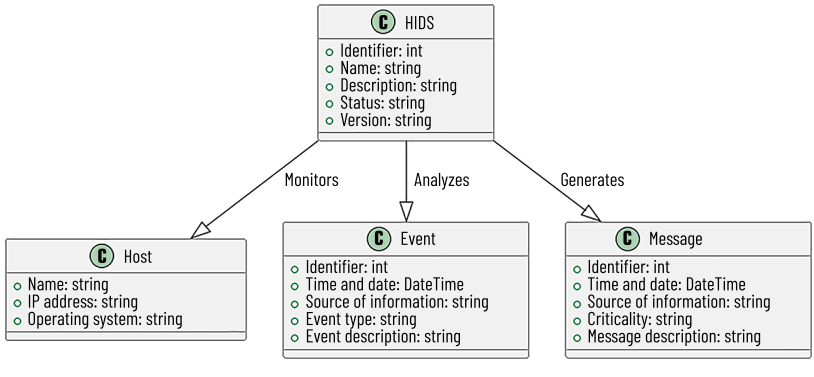


Fig. 3.2 HIDS operation principle

SIEM (Security Event Management) systems are systems that provide centralized log processing by collecting logs (primarily security-related) from various network devices and applications, and by analyzing and storing these logs. If the system detects an attack, it can respond through its incident management channels, which includes notifying personnel and initiating countermeasures. SIEM can also help an organization comply with data retention policies, where the latter can be useful in cases of electronic incident detection (also known as litigation preparation) and forensics [105, 106].

It is worth noting that having a SIEM system in addition to a NIDS offers several key advantages. A SIEM serves as a centralized platform for collecting, correlating, and analyzing security data from multiple sources, including NIDS and HIDS. This provides a complete picture of the organization's security posture, enabling real-time threat detection, cyberattack response, and forensic analysis of cybercrime. In addition, SIEM's advanced analytics and correlation capabilities help security teams identify complex attack patterns and effectively prioritize alerts, reducing the risk of false positives and alert fatigue. In addition, SIEM's reporting and compliance capabilities facilitate regulatory compliance and provide valuable information for security improvements. Ultimately, combining SIEM with NIDS and HIDS increases the organization's overall cybersecurity resilience, providing a classic approach to addressing cyberthreats. A SIEM system is schematically represented in **Fig. 3.3** appended [83-93].

This schematic image defines three rectangular SIEM containers, Security Data Sources, and External Data Sources. Inside the SIEM container, the components of the SIEM system are listed. The arrows indicate the connection between the SIEM system, security data sources, and external data sources. According to the defined functions of the SIEM system, it is determined that it is a standalone solution that can already be used to detect cyberattacks. However, SIEM systems usually do not offer organizations mechanisms to respond to threats or information security incidents [105].

For such purposes, next-generation firewalls and endpoint intrusion detection and response systems can be used at the network level.

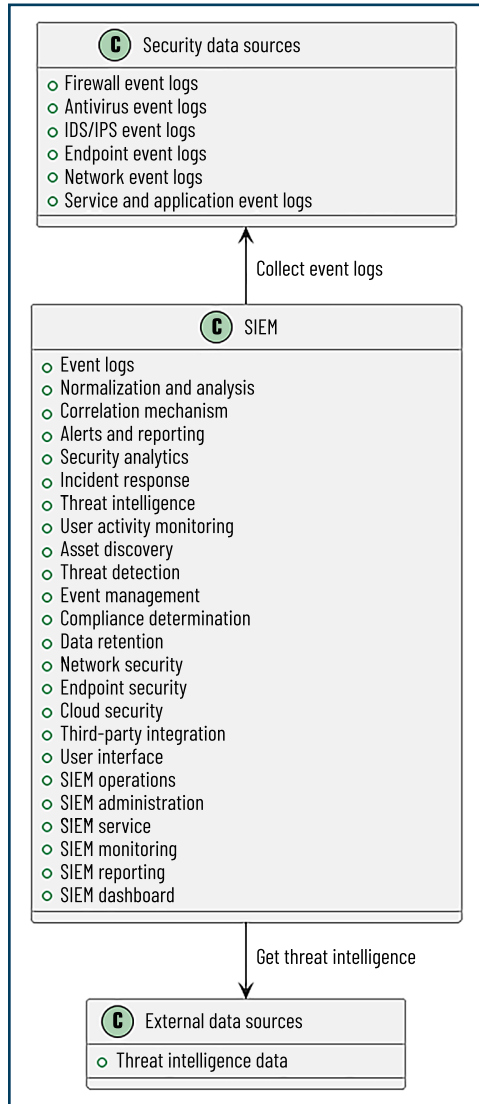


Fig. 3.3 SIEM operation principle

Endpoint detection and response (EDR) methods are designed to overcome the shortcomings of behavior-based detection methods. Because EDRs support actions across operating systems and are developed using open source, they allow experts around the world to collaborate and prepare responses faster than the

rate of attack evolution [83, 94, 96, 105]. The EDR principle is presented in **Fig. 3.4**, which defines three rectangular containers: EDR System, Endpoints, and Central Server. The EDR System container represents the main components of the EDR system. The Endpoints container represents the endpoints (devices) that are monitored. The Central Server container represents the central server that manages the endpoint agents and stores data. In this diagram, the EDR system contains intrusion detection mechanisms and reports on these mechanisms to the system user [94, 98, 99, 105].

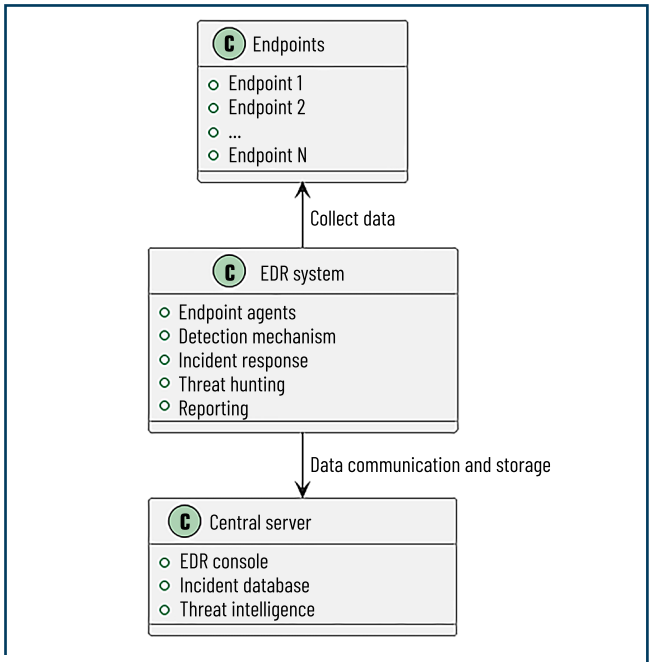


Fig. 3.4 EDR operation principle

The schematics of EDR and SIEM illustrate the use of information security threat intelligence. Integrating threat intelligence into EDR and SIEM systems is essential to stay ahead of evolving cyber threats, improve the accuracy of threat detection, and enable faster incident response [83–88, 96, 102, 105–110]. Cyber threat intelligence goes through a life cycle called the intelligence life cycle. The intelligence life cycle is the process of discovering, collecting, and developing raw data and information into intelligence that is used by decision makers [96–105, 111–113]. When this process is done correctly, intelligence activities can be performed in a focused and well-coordinated manner to meet user needs. The intelligence is collected in a threat intelligence system (TIS) – a cybersecurity solution designed to collect, analyze, and disseminate relevant information about cybersecurity threats and vulnerabilities. The system collects data from a variety

of sources, such as open source intelligence, private channels, and internal security logs, and then processes this data to provide organizations with actionable information about potential threats [96–105]. Threat intelligence systems help security teams adapt to new threats, tactics, and methods used by cybercriminals. These systems play a critical role in improving an organization's cybersecurity posture by providing timely and contextual information that helps detect threats, respond to incidents, and make decisions. A schematic representation of a threat intelligence system is shown in **Fig. 3.5**. It defines two rectangular containers: Threat Intelligence System and External Data Sources. The Threat Intelligence System container represents the main components of a threat analysis system. The External Data Sources container represents external sources of threat data, such as public and private threat feeds and data providers [106, 107].

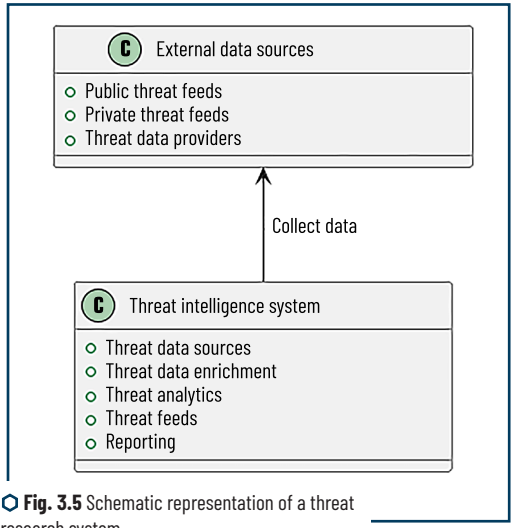


Fig. 3.5 Schematic representation of a threat research system

As already mentioned, cloud systems such as Amazon GuardDuty and Azure Log Analytics can be used for cybercrime analysis. They offer advanced capabilities for monitoring and analyzing logs, events, and network traffic and API calls, which allows organizations to proactively detect and investigate cyber threats in the cloud infrastructure [83, 97, 105, 112]. A schematic representation of the principle of operation of such systems is shown in **Fig. 3.6**.

The schematic representation defines three rectangular containers: Investigation System, Cloud Environment, and Third-Party Data Sources. The Investigation System container lists typical components of a dedicated cloud-based cybercrime investigation system [83–110]. Arrows indicate the connection between the investigation system and the cloud environment (collected data) and external data sources (imported logs and threat data). By integrating specialized cloud systems into their security strategies, organizations can maintain robust security, comply with international standards, and reduce the risk of data leakage [107].

The importance of integrating various cybersecurity systems such as EDR, SIEM, threat intelligence systems, and NIDS/HIDS for cybercrime analysis for the components of the information systems infrastructure lies in their joint ability to create a comprehensive, multi-layered cybercrime defense strategy. These systems work together to enable organizations to detect and respond to cyberthreats. EDR provides real-time visibility into endpoint activity, SIEM collects and correlates data from across the organization, threat intelligence systems provide timely threat context, and NIDS/HIDS monitor network and host behavior [18, 83, 97, 105, 106]. Together, they offer holistic threat detection, early detection capabilities, rapid response, and informed decision-making during cybercrime investigations. This integrated approach strengthens an organization's resilience against cybercriminals by reducing vulnerability, minimizing losses, and protecting critical assets [112, 115, 120]. Modern security systems provide organizations with the capabilities to detect and respond to security threats. However, they also have their own set of challenges and issues:

1. Depending on the product and type of deployment (on-premises or cloud), the cost can vary, and commercial solutions often offer high-cost SIEM solutions [60, 89, 113].

2. SIEM systems collect and analyze a huge amount of security data from various sources. This data overload can lead to a large number of alerts, making it difficult for security teams to prioritize and respond effectively. Companies should strive to implement SIEMs that, by analyzing hundreds of millions of event log files, identify 2–3 actionable alerts with context. A SIEM without any optimization will generate 1000 alerts [61, 108].

3. EDR solutions generate a significant amount of endpoint data, including logs, events, and telemetry. Analyzing this data can be resource-intensive and generate a large number of alerts. Threat intelligence systems can provide a vast amount of data about emerging threats and vulnerabilities, but it can be difficult for organizations to filter and effectively apply this data to their environment [62, 115–119].

4. Given the previously stated number of investigations that the average SOC analyst can perform, it is reasonable to assume that a company with 100 million log files per day would require approximately 100 data analysts. The high volume of alerts from SIEM systems can lead to alert fatigue, where security analysts are overwhelmed by the sheer volume of alerts, resulting in missed or delayed responses to critical threats. EDR solutions can generate numerous alerts, many of which may be false positives. This can cause security organizations to become desensitized to alerts and miss real threats [102, 120–122].

5. SIEM systems can generate false positive alerts due to misconfigured rules or inadequate correlation. This wastes time and resources investigating non-existent threats. Considering EDR solutions, they can trigger alerts based on suspicious behavior that may sometimes be normal activity [97, 110, 123].

6. SIEM and EDR systems are often limited in the types of events they can investigate, and they often do not support all types of event sources and services that can send data to a SIEM. Integrating threat intelligence feeds into existing security systems can be complex and time-consuming. This requires careful tuning to ensure that relevant threat intelligence is used effectively. Also, without a vulnerability assessment process, having a threat intelligence system may not be practical, as vulnerability assessment plays a key role in the development of information systems. However, there are many different vulnerability assessment tools and methods to choose from, and there is little information about which vulnerability assessment methods to use and when [115].

7. Managing security in a cloud infrastructure often involves the use of multiple cloud providers and tools. Integrating these disparate systems can require additional resources.

8. Threat actors are constantly adapting and changing tactics, making it difficult for organizations to keep up with new threats. Threat intelligence channels must be constantly updated and improved. Cloud environments are attractive targets for attackers, and new attack vectors continue to emerge. Organizations must continually adapt their security measures.

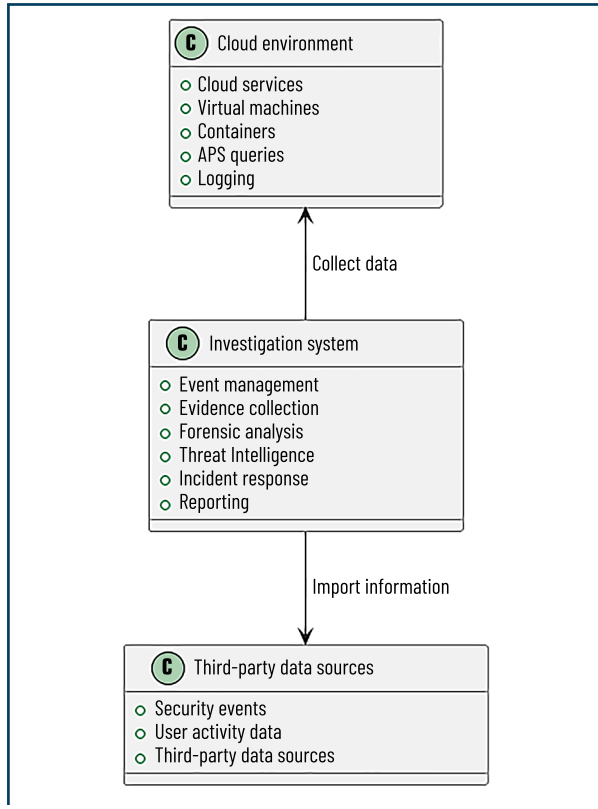


Fig. 3.6 Schematic representation of a cloud-based cybercrime investigation system

In summary, while modern security systems offer powerful capabilities, they also pose challenges related to data management, notification overload, integration complexity, privacy, compliance, and the ever-changing threat landscape. Organizations must address these challenges to maximize the effectiveness of their security operations and protect against new threats. To improve the process of threat analysis and

the detection of threats specific to the information system, a DevSecOps process for continuous security testing can be implemented, and it is worth noting that one of the most important attributes of any security testing is coverage. To assess the security of an information system, an automated scanner must be able to accurately interpret the program [47, 120, 123–130].

To improve cybercrime investigation processes, solutions such as SIEM systems, EDR and cloud security analysis systems can use machine learning algorithms. Artificial intelligence methods based on security analysis modeling can be used to solve various cybersecurity problems and tasks, such as automatic identification of malicious actions, phishing detection, malware detection, cyberattack prediction, fraud detection, access control management, anomaly or intrusion detection, etc. A corresponding analysis of the capabilities of artificial intelligence algorithms in the field of cybercrime investigation is described in the next section [108, 127, 131–136].

3.2 USING ARTIFICIAL INTELLIGENCE ALGORITHMS TO INVESTIGATE INFORMATION SECURITY EVENTS AND INCIDENTS

An anomaly is an irregular data point generated by a process different from that which generated the rest of the data [94, 96, 97]. AI is an emerging technological science that studies and develops theories, methods, techniques, and programs that simulate, extend, and augment human intelligence [88, 105, 110]. Machine learning (ML) algorithms are a branch of artificial intelligence that is closely related to (and often overlaps with) computational statistics, which also focuses on making predictions. ML has close ties to mathematical optimization, which provides methods, theory, and applications [83, 98, 102]. ML is often used in cybersecurity. Its ability to analyze data, discover patterns, and make predictions can be used to detect anomalies, classify security events and incidents, analyze text data, and monitor behavior in real time. When integrated into security systems, it significantly improves an organization's ability to detect and respond to security threats, ultimately strengthening cybersecurity [94, 96, 105].

One of the key applications of ML algorithms in security is anomaly detection. ML models can be trained to detect normal patterns of behavior in an information system. Any deviations from the normal patterns of behavior are flagged as anomalies. For example, unusual query usage, network traffic spikes, or unexpected resource consumption can trigger alerts [83, 94]. ML algorithms can also classify security events as benign or malicious. They can analyze various data sources, such as security logs, network traffic, or user behavior, to determine the nature of the events. For example, ML models can distinguish between legitimate emails and phishing attempts based on analysis of email content, sender behavior, and historical threat data [97, 102, 105].

One popular machine learning algorithm used in cybersecurity is the Isolation Forest due to its ability to detect anomalies. Isolation Forest does not use distance or density to detect anomalies, avoiding the computationally intensive distance and density methods [64, 65, 78, 129]. Its efficiency and scalability make it well suited to processing large amounts of log data, which is an important aspect of timely threat detection, as recent IBM studies have shown that it takes 208 days to detect an information security incident [126, 127, 135–137]. Unlike some other algorithms, Isolation Forest is less sensitive to the underlying data distribution,

making it adaptable to the diverse and changing attack patterns seen in cybersecurity. Furthermore, it is a one-class learning algorithm, meaning that it can learn normal event characteristics without requiring labeled anomalies during training [65, 78, 137].

Another type of model often used by information security professionals is deep learning, which uses neural networks with multiple layers to analyze and interpret complex data. In the context of cybersecurity, deep learning models have proven to be highly effective in solving complex and diverse security problems [71]. They are particularly well suited for image-based security tasks, network traffic analysis, and time series data processing. One of the main applications of deep learning in security is the use of convolutional neural networks (CNNs) for image-based security tasks. CNNs are adept at analyzing images and videos to detect potential security threats. For example, they can be used in surveillance systems to recognize unauthorized persons or objects in restricted areas, enhancing physical security [63, 72]. In contrast to CNNs, recurrent neural networks (RNNs), a type of deep learning model, are valuable for sequence-based data analysis. They are used in security for tasks such as network traffic analysis, where they can identify patterns in data streams that may indicate security breaches or suspicious activity. RNNs are also effective for analyzing time series data to detect anomalous behavior [71–73].

Deep learning models, including CNNs and RNNs, are used in real-time intrusion detection systems. These systems monitor network traffic, identifying patterns associated with known and emerging threats. They can quickly send alerts and trigger automated responses to mitigate attacks, such as blocking malicious IP addresses or quarantining compromised devices. Deep learning is often used in antivirus solutions to improve malware detection. They can also be used for user and entity behavior analysis (UEBA) to track the behavior of users and entities on networks. They detect deviations from established norms, signaling potential insider threats or compromised accounts. Thus, Deep Learning is an important tool in cybersecurity, particularly in security events and incident investigation. Although deep learning has achieved great success in transforming many data mining and machine learning tasks, popular deep learning methods are not suitable for anomaly detection due to some unique characteristics of anomalies, such as: rarity, heterogeneity, and prohibitively high cost of collecting big data [68]. Also, privacy and security issues of DL have been identified, such as the DL model can be stolen or reverse-engineered, sensitive training data can be obtained, and even a recognizable facial image of the victim can be recovered. In addition, recent works have found that the DL model is vulnerable to adversarial examples corrupted by unobservable noise, which can lead the DL model to make incorrect predictions [66, 68].

Random Forest is a versatile machine learning algorithm that plays a key role in cybersecurity incident and event analysis. Random Forest model is essentially a learning process for classification, regression, and other tasks. More specifically, the RF model is based on decision trees and a bundling mechanism (i.e., bootstrap aggregation) to avoid the problem of overfitting complex decision trees [64, 78]. Its ability to handle structured and unstructured data, combined with its efficiency in detecting patterns and anomalies, makes it a valuable asset for strengthening digital security and responding to security incidents. This model excels at detecting anomalies in event data. By studying historical data patterns, it can recognize deviations from the norm. During a security incident, not all alerts need to be analyzed equally, and the Random Forest algorithm can prioritize alerts based on their severity and potential impact. By assigning

risk scores to events, it helps security teams respond to incidents to focus their efforts on the most critical threats first. The model can also analyze the behavior of users and objects on a network or system. Random Forest can detect unusual user behavior, such as repeated failed login attempts or unauthorized access to sensitive resources. This behavioral analysis helps to accurately identify internal threats and external attacks.

Therefore, the Random Forest model can be used in security incident and event analysis, enabling organizations to detect, assess, and respond to security incidents.

To determine the algorithm that is best suited for detecting information security anomalies, the models were tested against each other. For comparison, the following experiments were conducted:

1. Three anomaly detection models were created: Random Forest and Isolation Forest written in Python using libraries such as Pandas for data processing and Scikit-learn for machine learning tasks, which provides the implementation of the Random Forest algorithm and tools for model evaluation. The Deep Learning model was designed to detect anomalies using a multi-layer perceptron (MLP) classifier, a type of neural network implemented in the scikit-learn library.

2. Two arbitrary datasets were created to train the three models. The first is an arbitrary dataset. The second is an arbitrary set of events from the NGINX web server with scan attack attributes.

3. Each model was tested on the same dataset: an event log with scan attack attributes.

4. The following metrics were used to evaluate the model: precision, accuracy, recall, *F*1 score, and area under the ROC curve (AUC-ROC) [74].

Accuracy is the ratio of correctly identified predictions to the total number of predictions, this metric is calculated as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{3.1}$$

where *TP* – true positives, *TN* – true negatives, *FP* – false positives, *FN* – false negatives.

Precision is the ratio of true positives to the sum of false and true positives. The metric is described by the following formula:

$$Precision = \frac{TP}{TP + FP}. \tag{3.2}$$

Recall is a metric that indicates the ability of a model to find all relevant cases (positives) in a dataset. This metric is calculated as:

$$Recall = \frac{TP}{TP + FN}. \tag{3.3}$$

The *F*1 score is the harmonic mean of precision and recall. It is a way of combining both metrics into a single score that captures both false positives and false negatives. This metric is calculated using the following formula:

$$F1Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}. \quad (3.4)$$

Also, an important metric is the area under the ROC curve (AUC-ROC). The AUC-ROC curve is a measure of performance for classification problems at different thresholds. ROC is a probability curve, and AUC is a measure or measure of resolution. AUC indicates how well a model is able to distinguish between classes. The higher the AUC, the better the model is at predicting 0 classes as 0 and 1 class as 1. An AUC score of 0.5 indicates that there is no random guessing. Although AUC-ROC is a graphical representation and does not have a simple formula, the general idea is to calculate the area under the ROC curve, which is a plot of true positive rate (TPR) versus false positive rate (FPR) at different thresholds.

$$TPR = \frac{TP}{TP + FN}; \quad (3.5)$$

$$FPR = \frac{FP}{FP + TN}. \quad (3.6)$$

The results of model testing are shown in **Fig. 3.7**.

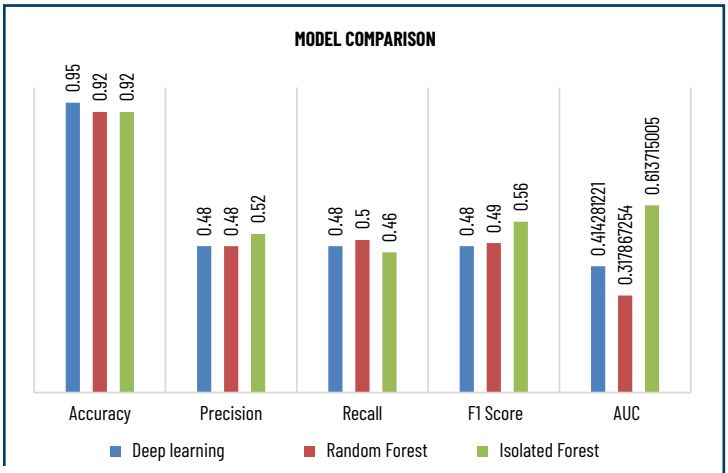


Fig. 3.7 Model comparison

The results of this experiment show that when detecting anomalies from a dataset of information security events generated by a scanning attack on the NGINX service, the deep learning model shows moderate performance with the lowest AUC, indicating problems with class discrimination. Random Forest

model: shows high accuracy, but its precision and *F*1 score are lower compared to other models, and the AUC score is the worst among the three models. This indicates difficulties in effective anomaly detection.

Isolation Forest model: shows high accuracy and precision and the highest AUC and *F*1 scores, indicating the lowest false positive rate.

Also, as part of the model comparison, an analysis of the literature on the topic of machine learning algorithms was conducted and **Table 3.1** was formed, which uses standard criteria for machine learning algorithms to form a comparative characteristic: complexity, ability to work with different types of data, ability to detect anomalies, ability to interpret results, size of training data, scalability, real-time data processing, presence of false positives and range of application.

Table 3.1 Comparative characteristics of machine learning algorithms for anomaly analysis

Criterion	Deep learning	Isolation Forest algorithm	Random Forest algorithm
Complexity	Deep and complex neural networks with many parameters	A simple and efficient decision tree algorithm	An ensemble of decision trees offering moderate complexity
Ability to work with different types of data	Effective for unstructured data	Suitable for structured and unstructured data, including tabular and log data	Generally suitable for structured data, log analysis, and event data
Ability to detect anomalies	Effective for detecting sequence-based anomalies, but may require large datasets	Able to detect anomalies, especially from large datasets	Able to detect anomalies but may require feature development
Ability to interpret results	Often considered a "black box" with limited interpretation for decision-making	Provides some interpretation through feature importance evaluation	Offers moderate interpretation through feature importance analysis
Size of training data	Effective training requires large amounts of labeled data, which can be difficult to obtain in cybersecurity	Requires less labeled data for training	Typically requires moderate amounts of labeled data for training
Scalability	Can be computationally intensive, especially for deep architectures	Scalable and efficient for large datasets	Scalable, but performance can degrade with very large datasets
Real-time processing	Can be difficult to process in real-time due to computational requirements	Good for real-time processing and fast anomaly detection	Suitable for real-time or near-real-time processing in many cases
Presence of false positives	Prone to higher false positives due to complexity	Known for its ability to reduce false positives	Offers moderate false positive rate depending on setup
Range of application	Effective for image analysis, natural language processing, and complex pattern recognition	Very good for anomaly detection in structured and unstructured data	Generally suitable for a wide range of cybersecurity tasks, including attack detection and event log analysis

Therefore, based on the above analysis, the Isolation Forest model offers the following advantages over Random Forest and deep learning algorithms for anomaly detection, which can be the basis for analyzing information security events and incidents:

1. Efficiency and rate: Isolation Forest can efficiently process large amounts of data and quickly detect anomalies.

2. Scalability: Isolation Forest has high scalability and the ability to process large data.

3. Robustness to data distribution: Isolation Forest is less sensitive to the underlying data distribution. It works well with both skewed and multimodal data distributions, making it adaptable to various cybersecurity datasets. In contrast, Random Forest and Deep Learning methods may require additional data preprocessing and tuning to effectively handle various data distributions.

4. Single-class learning: Isolation Forest is a single-class learning algorithm, meaning it can learn from the characteristics of regular data without the need for labeled anomalies during training. This makes it suitable for analyzing new attack models.

5. Ease of implementation: Isolation Forest is relatively easy to implement and does not require significant hyperparameter tuning, making it an attractive choice for rapid prototyping and deployment.

While Isolation Forest offers these advantages for many cybersecurity anomaly detection scenarios, it is important to note that the choice of algorithm should always be based on the specific requirements and characteristics of the data, as well as the cybersecurity task at hand. In some cases, Random Forest or Deep Learning methods may be more suitable, especially for tasks that involve complex patterns, image analysis, or natural language processing. Therefore, choosing the best algorithm depends on the specific context and goals of the cybersecurity task.

3.3 EXPLORING THE POSSIBILITIES OF USING CHATBOTS USING THE GPT MODEL FOR EVENT LOG ANALYSIS

Generative Pre-trained Transformer (GPT) is a natural language processing algorithm created by the American company OpenAI [79, 106, 122].

The main feature of this algorithm is its ability to memorize and analyze information. Thanks to its natural language processing capabilities, GPT can understand, classify, and analyze event logs. Thanks to its ability to analyze event logs, GPT can detect anomalies, provide insight into system behavior, and generate reports. This contributes to faster problem detection, root cause analysis, and helps automate the monitoring process [106, 129–132].

To study the capabilities of chatbots using GPT, a vulnerable environment was used, the main component of which is OWASP Juice Shop, installed on an Ubuntu server in the Digital Ocean cloud environment. A detailed diagram of the environment setup is shown in **Fig. 3.8**.

The following steps were taken to prepare the environment:

1. An Ubuntu server was installed in Digital Ocean, an Owasp Juice Shop container was installed, and Nginx was configured as a Reverse Proxy to record all requests to the web server.

2. The following applications were used to create event logs with potential attacks on the configured environment: Nessus, nmap, BurpSuite, and Metaspoit. Automated vulnerability scanning tools were used to cover as many types of attacks as possible.

3. To study various types of cyberattacks using GPT 3.5 and GPT 4, event logs generated after scans and exploitation of vulnerabilities were used, and an implementation of the GPT algorithm by OpenAI – ChatGPT was proposed.

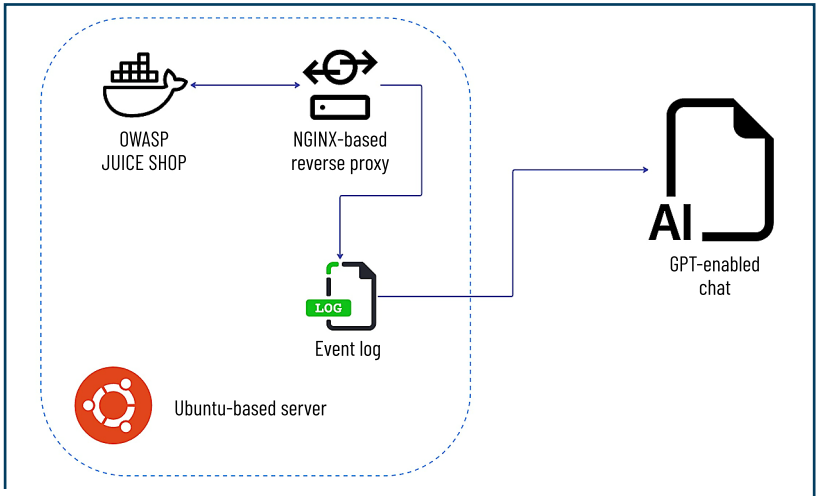


Fig. 3.8 Scheme of a potentially vulnerable environment

During the first experiment, the GPT 3.5 and GPT 4.0 algorithms were provided with event logs containing a directory traversal attack for analysis. A directory traversal attack, also known as a path traversal attack or directory hopping attack, is a type of cyberattack that exploits vulnerabilities in a web application's file handling mechanisms [81]. The attacker seeks to gain unauthorized access to the application's file system, allowing it to read, modify, or delete sensitive files and data. All queries in this experiment are in English.

Both GPT models were provided with 200 identical event logs: 100 encoded and 100 normalized event logs. Each model was provided with 10 events every hour for 10 hours. An example of the event logs provided for analysis:

92.253.XX.XX – [13/Mar/2023:21:33:39 +0000] "GET...../etc/passwd (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)"HTTP/1.0" 404 564 "-" "Mozilla/4.0

92.253.xxx.xxx – [13/Mar/2023:21:33:39 +0000] "GET / %uff0e %uff0e/ %uff0e %uff0e/ %uff0e %uff0e/ %uff0e %uff0e/ %uff0e %uff0e/ %uff0e %uff0e/ %uff0e %uff0e/ %uff0e %uff0e/ %uff0e %uffoe/ %uff0e %uff0e/ %uff0e %uff0e/ %uff0e %uff0e/etc/passwd HTTP/1.0" 400 166 "-" "-"

The algorithm analyzed and described the entries made by the attacker. The average time for analyzing a request without encoding from 10 attempts GPT 3.5 – 7.9 seconds, encoded – 8.3 seconds. In the case of GPT 4.0 – 5.8 seconds, encoded – 6.2 seconds.

The next attack that ChatGPT proposed for analysis is cross-site scripting. Cross-site scripting (XSS) is an attack aimed at the vulnerability of a web application that allows attackers to execute malicious scripts on web pages that users open. This vulnerability occurs when an application accidentally includes unverified data (usually entered by the user) on a web page without appropriate validation, protection or encoding. As a result, the attacker's malicious script is run in the victim's web browser, which can lead to unauthorized access to data, account hacking or other malicious actions [82]. The GPT 4.0 and GPT 3.5 models were used to analyze an XSS attack. The models were given 100 identical event log records: 10 records every hour for 10 hours.

Example of an event log record:

92.253.xx.xx – - [13/Mar/2023:21:33:53 +0000] "GET /xmd79sr7.asp? <script>document.cookie= %22testmtbo=2804; %22</script> HTTP/1.0" 404 564 "-" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)"

ChatGPT analyzed the attack and XSS usage patterns. It was determined that the attacker intended to manipulate the application using JavaScript and URL parameters. The event logs detected GET requests directed to the file "xmd79sr7.asp" that were accompanied by URL parameters containing malicious JavaScript content. The models identified the following entry as malicious: "<script>document.cookie = %22testmtbo = 2804; %22</script>". ChatGPT also concluded that the attacker wanted the software to present and subsequently run this code in the user's browser. Further analysis by ChatGPT of the server response revealed that the response contained a status code of 404, which means "Not Found". This means that either the web server failed to process the request, or the existing server settings mitigate such threats. The average analysis time using GPT 4.0 out of 10 attempts was 12.3 seconds and GPT 3.5 – 17.5 seconds.

Also, within the framework of this study, the possibility of detecting a vulnerability scanning attack was analyzed. For this, a set of event log records was used – 10 identical records for each model, 10 every hour, which were provided for analysis to the GPT models, in particular, they contained the following records indicating vulnerability scanning:

92.253.xx.xx- – [13/Mar/2023:21:33:52 +0000] "POST /spipe?Source=nessus HTTP/1.0" 404 564 "-" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)"

92.253.xx.xx – - [13/Mar/2023:21:33:52 +0000] "POST /cgi-bin/mainfunction.cgi HTTP/1.0" 404 564 "-" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)"

92.253.xx.xx – - [13/Mar/2023:21:34:07 +0000] "POST /flex2gateway/http HTTP/1.0" 404.564 "-" "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)"

The study used the GPT 4.0 and GPT 3.5 models. ChatGPT assessed that the provided event logs indicate attempts by attackers to identify vulnerabilities in the web server using various queries. The average time to analyze the event logs was 12.9 seconds using GPT 4.0 and 18.3 seconds for GPT 3.5, with each query considered separately. It is worth noting that the queries were evaluated and the responses were provided in Ukrainian.

For further evaluation and comparison of the models, event log records obtained by the server during the attacker's attempt to exploit the Log4j vulnerability were presented. In particular, the event logs contained the following entries:

92.253.xx.xx [13/Mar/2023:21:39:23 +0000] "GET /wp-login.php HTTP/1.0" 404 162 "${jndi:ldap://log4shell-generic 8Vno2Ky4QW5hhAz16ZUW${lower:ten}. w.nessus.org/nessus}""${jndi:ldap://log4shell-generic-8Vno2Ky4QW5hhAz16ZUW$ {lower:ten}.w.nessus.org/nessus}"

92.253.xx.xx – - [13/Mar/2023:21:39:23 +0000] "GET /wp-login.php HTTP/1.0" 404 162"${jndi:ldap://log4shell-generic 8Vno2Ky4QW5hhAz16ZUW${lower:ten}.w. nessus.org/nessus}""${jndi:ldap://log4shell-generic 8Vno2Ky4QW5hhAz16ZUW${lower:ten}.w.nessus.org/nessus}"

92.253.xx.xx – - [13/Mar/2023:21:39:28 +0000] "GET /wwwadmin.cgi HTTP/1.0" 404 162 "${jndi:ldap://log4shell-generic-8Vno2Ky4QW5hhAz16ZUW${lower:ten} .w.nessus.org/nessus}""${jndi:ldap://log4shell- generic8Vno2Ky4QW5hhAz16ZUW$ {lower:ten}.w.nessus.org/nessus}"

Log4Shell (CVE-2021-44228) is a critical vulnerability discovered in the Apache Log4j library that allows an attacker to remotely execute malicious code [83]. Both models detected an attempt to exploit CVE-2021-44228. The average analysis time from 10 attempts conducted over 10 hours for GPT 4.0 is 15.3 seconds, and GPT 3.5 is 18.1 seconds.

Thus, as a result of these experiments, it was determined that from all the provided event log records, the GPT 3.5 and GPT 4.0 models processed and successfully detected all types of attacks, which was confirmed by experiments using event log arrays. For model comparison, the experimental results are shown in **Table 3.2**. For data analysis, the difference in analysis rate was determined in percentage and a diagram comparing the models was formed, shown in **Fig. 3.9**.

Table 3.2 Comparative characteristics of GPT 3.5 and GPT 4.0 for event analysis

Attack	Average analysis time from 10 attempts (GPT 3.5)	Average analysis time from 10 attempts (GPT 4.0)	Analysis rate difference %
Normalized Directory Traversal Attack	6.2	5.8	6.45 %
XSS Attack	6.4	5.7	10.94 %
Encoded Directory Traversal Attack	8.3	7.9	4.82 %
Vulnerability Scanning Attack	17.5	12.3	29.71 %
CVE-2021-44228 exploitation	18.1	15.3	15.47 %

GPT 4.0 generally demonstrates improved performance in handling and detecting various types of cyberattacks compared to GPT 3.5. Across all attack types tested, GPT 4.0 consistently demonstrates faster response times. Specifically, for vulnerability scanning attacks and exploits of CVE-2021-44228, GPT 4.0 is up to 29.71 % and 15.47 % faster at parsing event logs than GPT-3.5, respectively. This suggests that GPT 4.0 may be more effective at parsing more complex attack types. For web attacks such as directory traversal (both normalized and encoded) and XSS attacks, GPT 4.0 is up to 10.94 % faster.

The experiments conducted have established that GPT 4.0 not only accurately determines the type of cyberattack, but also provides overall faster detection, which can be crucial during real threats to the security of the information system, where response time needs to be minimized.

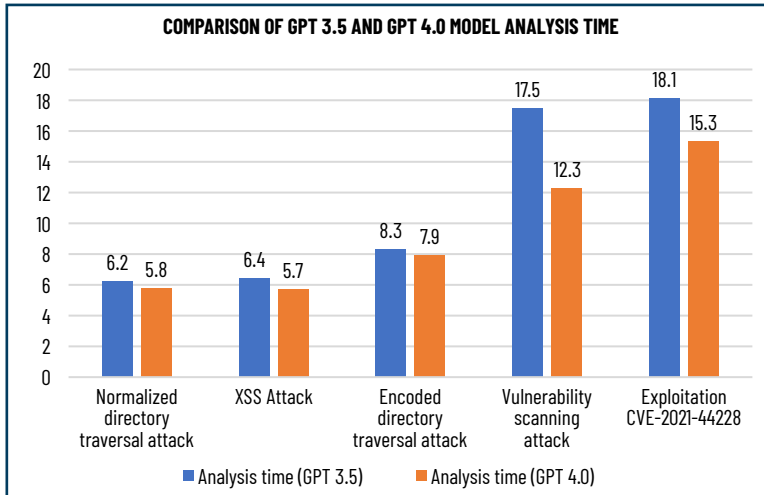


Fig. 3.9 Comparison of analysis time of GPT 3.5 and GPT 4.0 models

3.4 CYBERCRIME RESEARCH MODEL CONCEPT

After the research conducted in the previous sections, the concept of a cybercrime investigation system model was formed. This concept is presented in **Fig. 3.9** and consists of the following components:

1. The threat investigation system is an external system responsible for collecting indicators of system compromise (ISC), in particular malicious IPs, hash sums, domain names, emails. This information is transmitted to the event analysis component via a secure API, which allows the event analysis component to quickly identify and respond to potential threats.

2. The vulnerability management system is an external system responsible for consolidating information about vulnerabilities identified for the information system and sending this information to the event analysis component.

3. The static application security testing (SAST), dynamic application security testing (DAST), and software component analysis (SCA) components are external systems used by software developers to analyze software vulnerabilities. The event analysis component receives information from these systems.

4. GPT model – a third-party system, the connection to which is provided using the API interface. The event analysis component sends consolidated and masked information about a potential cybercrime to this system.

5. Anomaly detection component – an internal algorithm of the cybercrime investigation system, responsible for studying normal user activity by analyzing information system events and detecting unusual actions.

6. Event analysis component – an internal algorithm of the cybercrime investigation system, responsible for consolidating information from various data sources and analyzing events using third-party services with GPT support.

7. Data masking component – an internal algorithm responsible for masking data of the cybercrime investigation system.

8. Presentation component – an interface for working with the system, which is used by the user.

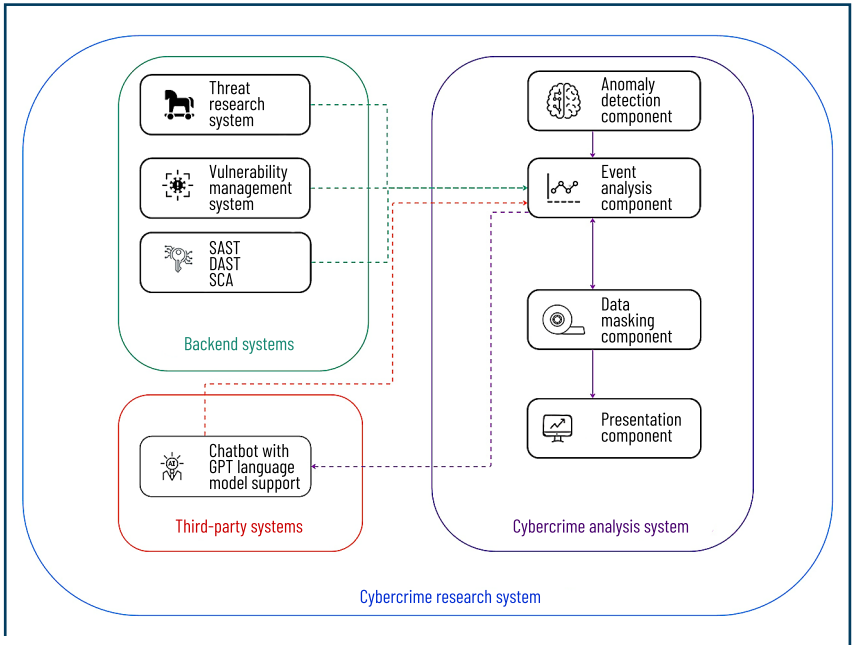


Fig. 3.10 Schematic representation of the components of the cybercrime investigation system

Therefore, all the necessary components were identified to build the cybercrime investigation system. A detailed description of the components and their purposes is provided in paragraph 6.4 of the work.

3.5 USING BLOCKCHAIN TECHNOLOGY TO INVESTIGATE CYBERCRIMES AT INFORMATION ACTIVITY FACILITIES

To collect event logs for the cybercrime investigation system, it is proposed to use a decoy system based on Blockchain technology. The system uses dynamic attributes of Blockchain technology to change the services used by nodes. The attacker's actions are recorded in the event log generated by the system [26–28]. The system is schematically indicated in **Fig. 3.11**, and a detailed description of the principle of the system's operation and event generation is given below.

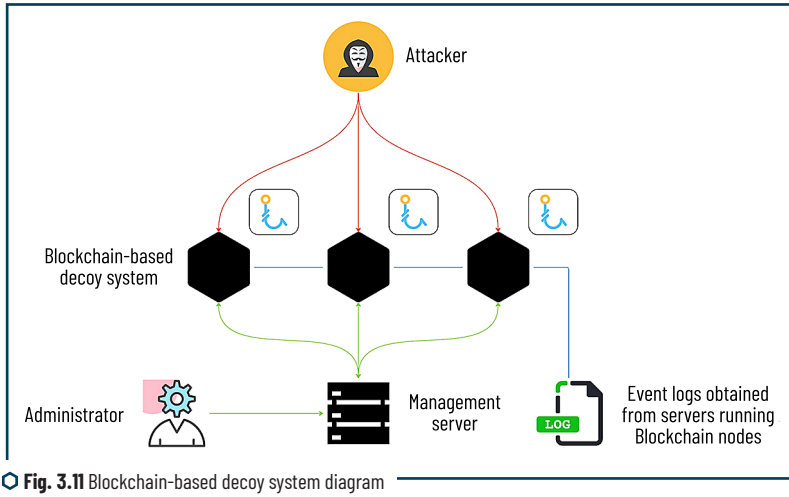


Fig. 3.11 Blockchain-based decoy system diagram

The system performs creation, sending, receiving, waiting, opening, closing, recording, restoring, and compromising. The notations used for the system are described in **Table 3.3**. The system actions and their parameters are summarized in **Table 3.4**.

Table 3.3 Notations used for the Blockchain-based decoy system

Name	Designation
1	2
System State	$\{st_n, st_c\}$
Data	$\{d_1, d_2, ..., d_n\}$
Malicious Data	$\{md_1, md_2, ..., md_n\}$
Data Channels	$\{c_1, c_2, ..., c_n\}$

Continuation of Table 3.3

1	2
Assets	$\{a_1, a_2, ..., a_n\}$
Services	$\{s_1, s_2, ..., s_n\}$
Event Log Entry	$\{log_1, log_2, ..., log_n\}$

Table 3.4 Description of system actions

Function	Description
generate(d)	asset creates data
send(d,ch)	asset transmits data over data channel
receive(d,ch)	asset receives data over data channel
deny(d,ch)	asset rejects receive data over data channel
compromise()	asset is compromised
record(log)	asset creates event log entry
await(answer)	asset waits for response after sending

The method of collecting event logs from Blockchain-based decoys is described below:

1. A set of services $\{s_1, s_2, ..., s_n\}$ of an asset receives requests from an attacker. The generate(d) and send(d) functions represent the manipulation of the services' output data, and the receive(d) function represents their receipt from the outside. After receiving the request, the asset can operate in either normal or compromised mode. Normal mode means that the attacker has not affected the asset and the asset maintains normal operation. However, compromised mode indicates that the attacker has gained unauthorized access to the asset. The states $\{st_n, st_c\}$ are divided into two categories: st_n for normal mode and st_c for compromised mode. It is worth noting that the data transmitted by the assets guarantees the normal operation of the system, and they play an important role in security analysis [85]. If an asset receives malicious data from an attacker, it becomes compromised:

$$\{a_1, a_2, ..., a_n\}, receive(md) \rightarrow compromise\ \{a_1, a_2, ..., a_n\} \wedge st_n, \{a_1, a_2, ..., a_n\}.$$

2. It is assumed that if a normal asset receives malicious data, it will enter a compromised mode, further compromising itself. To prevent a compromised asset from intercepting data between other assets, the data channels $\{s_1, s_2, ..., s_n\}$ must be protected [86].

3. All states in the assets illustrate the overall state of the system, that is, the continuous authorized behavior of each asset ensures the normal functioning of the system. Any assets communicate with each

other by sending and receiving data through communication channels. This indicates that sharing a single channel allows both connecting and transmitting shared data. Data exchange can only occur when connected through a single communication channel [87]. So, it is possible to define the following statement: an asset generates data and sends data to an asset via a communication channel:

$$a_1 \rightarrow generate(d_1) \wedge send(d_1).$$

4. After receiving data from asset a_1, asset a_2 can decide whether to accept or reject this data. When asset a_2 receives and accepts malicious data, it becomes compromised [88]:

$$a_1, send(c, d_1) \rightarrow a_2, receive(c, d_1),$$
$$a_1, send(c, d_1) \rightarrow a_2, deny(ch, d_1).$$

If: $d_1 = md$;
Then: a_1, *compromise*;
So: $sc(a_1)$.

5. The normal asset is a legitimate part of the system and ensures the normal functioning of its services for users. The system adapts and focuses on transmitting data for further attack analysis and records all attack data in the event log [89]:

$$a_2 \rightarrow record(\text{log}) \wedge await(response).$$

The event logs generated by the Blockchain-based decoy system are used in this study for event analysis by the cybercrime investigation system.

4 DEVELOPMENT OF A METHOD FOR USING SOFTWARE DECOYS AS ELEMENTS OF PROTECTION OF COMPUTER NETWORKS OF INFORMATION ACTIVITY OBJECTS BASED ON BLOCKCHAIN TECHNOLOGY

4.1 DEVELOPMENT OF A DECOY-BASED MODEL OF DECENTRALIZED COMMUNICATION

The analysis of deception systems (Deception) and Honeypot in Section 1 showed the prospects for development and evolution of this technology and its possibilities for expansion. However, both Deception and Honeypot are centralized systems that still have all the disadvantages of a centralized approach, namely the management server. When the main link is detected, the hacker can adjust his/her actions. This risk can be mitigated by building a protection system that does not depend on only one central node [82]. Blockchain is a multi-node system in which each node must confirm the information coming to one of the links before letting it into the general data flow [36, 45, 60]. The property of dynamic change and validation of Blockchain nodes can significantly strengthen protection systems and prevent the problem of centralized control. Based on this, it is necessary to model a dynamic distributed control system using the dynamic properties of Blockchain and explore the parameters of this system. Therefore, let's introduce a dynamic distributed model of a software decoy formed by *N* hosts and four services.

As shown in **Fig. 4.1**, there are two participants: a hacker and a legitimate user who is synchronized with the real service (i.e. the client can store the location using the real service and knows the exact location). The *N* hosts constitute a private Blockchain, which is a P2P network and does not open its doors to the outside world.

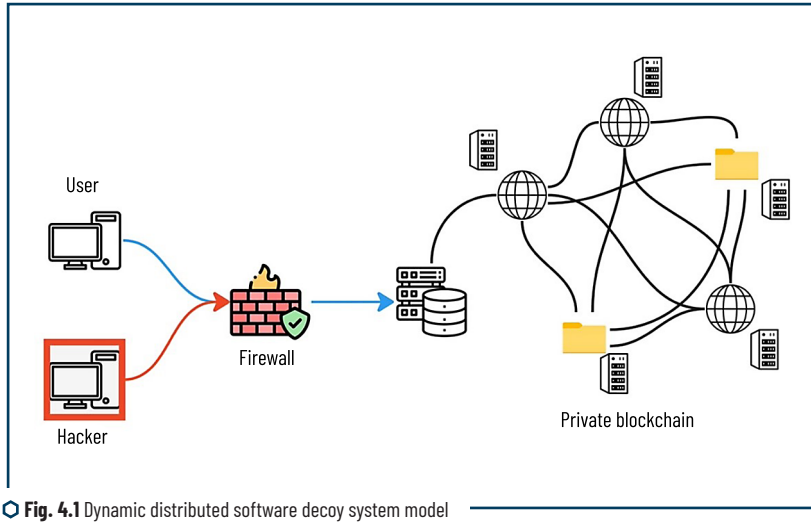


Fig. 4.1 Dynamic distributed software decoy system model

Solana (i.e., Blockchain platform) serves as the bottom layer in the system. *N* hosts constitute a private Blockchain, which forms a P2P network. By calculating the hash value of a block, a host in the private chain can mine a potential block and upload it to the chain. This mechanism guarantees the distribution and decentralization of the deployment architecture [60]. This process generally creates the concept of "floating hosts". Floating hosts are the process of constantly changing the temporary master host. The temporary master host executes the service distribution algorithm and sends the corresponding encrypted information to other hosts. The temporary master host executes the service distribution algorithm and sends the corresponding encrypted information to other hosts. As shown in **Fig. 4.2**, in the system, the block miner $Host_0$ (i.e., the host that successfully calculates a certain hash) becomes the master host in the period $Table_0$, and another host $Host_1$ can replace $Host_0$ in the next round. The host with more computing power is likely to be the temporary hub controller. If a narrowly configured host suffers from attacks and its performance degrades, it cannot serve as the hub host due to lack of sufficient computing power, and other hosts will automatically replace it. Therefore, the failure of the main host $Host_0$ does not matter to the entire system (i.e. the system functions normally). Attack logs recorded by one host are uploaded to the Blockchain, and other nodes synchronize these logs on our private chain. In this way, each node has complete data stored in a safe and secure manner for further forensics of attacks.

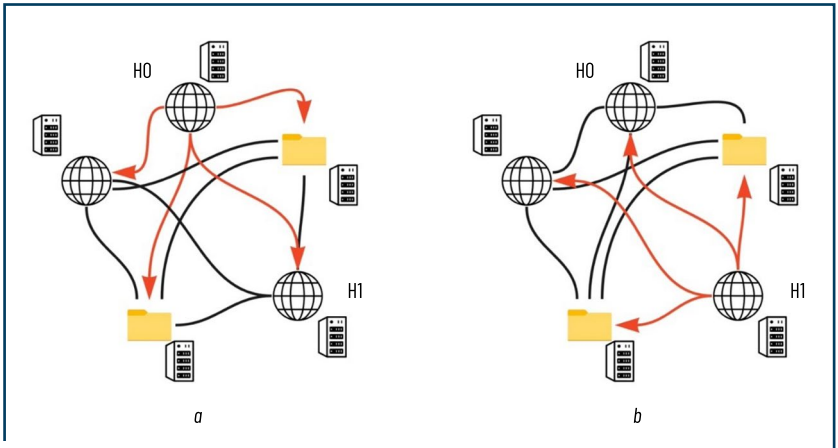


Fig. 4.2 Different main hosts: *a* – main host in $Table_0$; *b* – main host in $Table_1$

4.1.1 DESCRIPTION OF HOST COMMUNICATION IN THE BUILT BLOCKCHAIN NETWORK

The host that mines the block acts as a non-permanent centering controller. This central host generates conversion information that assigns each host to run different services (i.e., to run a real service or a decoy service) according to a random generation algorithm. The data contains service numbers and

the encoding 01, which will be encrypted using the 2048-bit RSA encryption algorithm [35, 60, 84]. The encrypted data is then sent to other hosts, the temporary center host in this private network. Upon arrival at the corresponding host, the information is decrypted and the plaintext is obtained. For encoding 01, zero is the symbol for starting the decoy service, and one represents the real service. A bitwise comparison is performed using the text, then the specified service is started to complete the execution procedure. For the authorized user, synchronization is performed to maintain normal operation. By sending the user encrypted information of the real service, the server can provide a regular service. In addition, the user can send the encrypted query data "whois + server name" to actively obtain the desired address of a specific service. In this way, the real user can access real system resources when using the service.

The formal description of the decentralized communication mechanism in **Fig. 4.3** looks like this:

1. At some point in time, the temporary master host $mHost_j$ requests a new coinbase from the Blockchain via the web3J interface. Coinbase represents the host that successfully mines the block. After that, $mHost_j$ generates a command called $Command_{update}$ to update it. $Command_{update}$ has a specific format. $K_{public} = (E, N)$ and $Enc_1 = ((Command_{update})^E modN)$ are computed. The encrypted message Enc_1 is sent to every other host in the private Blockchain. Upon receiving the message, these hosts with $K_{private} = (D, N)$ will compute $Dec_1 = ((Enc_1)^D modN)$. After verifying Dec_1 in a specific format, coinbase will be updated on each host. The host associated with it acts as the new temporary master host. Meanwhile, j in $mHost_j$ changes to a new value.

2. The new master host $mHost_j$ is authorized to execute the distribution algorithm. Service numbers and 01 codes are generated, which direct other hosts to open or close. These are considered service codes. A $Command_{changeSrv}$ message is sent, which contains the service codes. Different hosts receive different $Command_{changeSrv}$ messages. $Enc_1 = \left(Command^E_{changeSrv} \bmod N\right)$ is computed and sent to $cHost_i$, which represents the common host. $cHost_i$ executes $Dec_2 = ((Enc_2)^D modN)$ and receives a simple message. Dec_2 is set, and the host will open and close the corresponding services.

3. The client host sends a request command to one of these servers. The $Request_{srv}$ command contains the message: "who is Apache". $Request_{srv}$ is encrypted as $enc_1 = ((Request_{srv})^e modn)$ with public key $k_{public} = (e, n)$ and forwarded to the server.

4. The server decrypts the message enc_1 via its private key $k_{private} = (d, n)$. $dec_1 = ((enc_1)^d modn)$ is output and verified. The server has a set of request messages $\{R_0, R_1, R_2, R_3\}$. If, $S = dec_1 \oplus R_a = 0,\ and \in [0,3]$ the requested IP address IP_r in $enc_1=((IP_r)^e modn)$ will be returned to the client and its IP address will be added to the main list $List_{client}=\{IP_{c1}, IP_{c2}, ..., IP_{cc}\}$. Otherwise, the value of dec_1 will be ignored.

5. After obtaining IP_r in $dec_2 = ((enc_2)^d modn)$, the client host will connect to this IP address to obtain real resources.

6. Through the variables in different periods $\{T_1, T_2, T_3, T_t\}$, the real IP address will be updated to IP_f. The host configured with IP_r sends $Update_{src}$ command to clients according to $List_{client}$. The updated IP_f is encrypted as $enc_3 = ((Update_{src})^e modn)$.

7. Calculate $dec_3 = ((enc_3)^d modn)$. There are four commands $\{C_0, C_1, C_2, C_3\}$ which follow a special format in clients. If $s = dec_3 \oplus C_a = 0,\ and \in [0,3]$, the client connects to the new IP_f. Periodically switching services, the aforementioned steps will be executed cyclically.

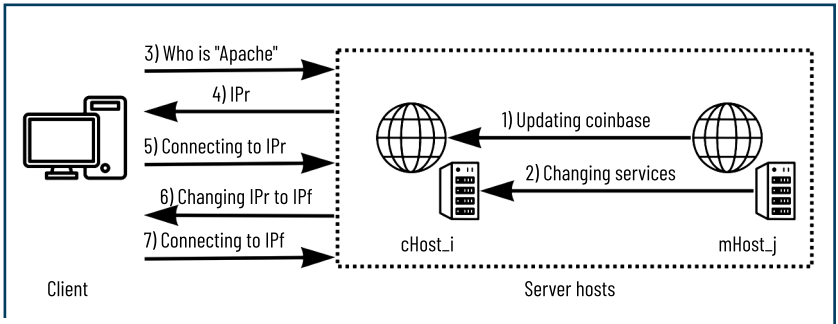


Fig. 4.3 Decentralized communication mechanism

4.1.2 TRANSFORMATION OF SERVICES DURING CONNECTION TO NODES

In this scheme, there are only four types of services, and each service has both real and fake attributes (i.e., four real services and four corresponding (fake) services). Periodic switching of services is performed every *Table* period. A comparison of the distribution of services is shown in **Fig. 4.4**. Both types of services are constantly changing [22].

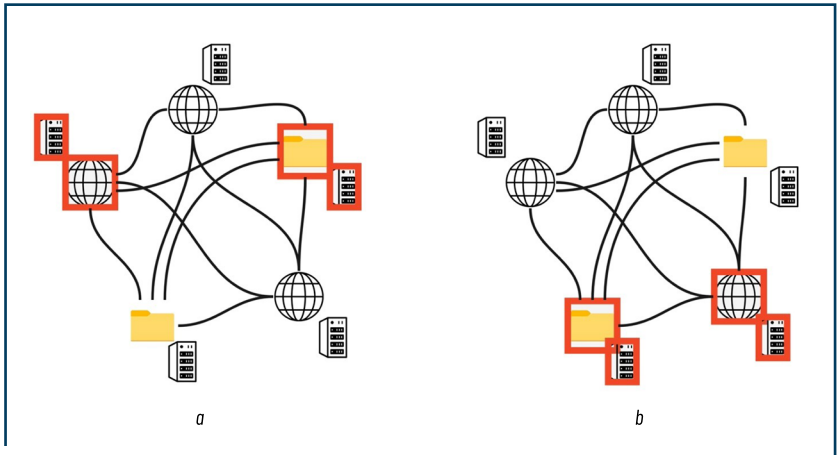


Fig. 4.4 Comparison: *a* – primary host in $Table_0$ before migration; *b* – primary host in $Table_1$ after migration

Based on the premise of the existence of anti-honeypot identification technology, there are three types of applications in the protection system:

1. If the service $Service_0$ on host $Host_0$ is real in $Table_0$, $Service_0$ can become fake in the next period $Table_1$. After the transformation, the attacker cannot access the real resources of the service in $Table_1$.

2. If the $Service_1$ service on $Host_1$ serves as a host service in $Table_0$, according to the promise of anti-honeypot technology, once the attacker discovers that the service is a trap, it will avoid $Service_1$, which can change to the real service in $Table_1$. Thus, it prevents the attacker from accessing real resources.

3. If the $Service_0$ service on $Host_0$ is real in $Table_1$, due to synchronization with real users, the client will only send requests to the real service. Since there are some fake services (such as honeypots), any access traffic to the honeypots $Service_0$, $Service_0$, ..., $Service_n$ in $Host_0$, $Host_1$, ..., $Host_m$ is marked as an attack record.

Thus, the transformation and relocation of services and services confuses attackers and protects the designed system.

4.2 DEVELOPMENT OF A METHOD FOR A DYNAMIC SYSTEM BUILT USING SOFTWARE DECOYS

This section presents a method for a dynamic system built using software decoys [86]. The basis of this method is the model and mathematical description presented in **Fig. 4.5** and the following steps:

1. When requests pass through the firewall, the software in the router classifies malicious requests and activates the process of deploying software decoys.

2. Malicious requests will be automatically redirected to the newly deployed decoy.

3. The Solana private Blockchain is created to exchange the deployment details of all services and decoys in the raised network.

4. A system is raised that dynamically deploys the required service on each node in a completely random manner. The service distribution program running on the temporary master server takes care of this functionality by taking the account address used by each server in the blockchain to access it as input and assigning a service code, which is a series of binary codes – digits, equal to the service number, including the individual decoy services. Example: Given four services such as Apache, MySQL and the corresponding decoy services, the service code is: 0000, where 0 means enabled and 1 means disabled.

5. The distribution program generates deployment details. These need to be securely shared between the temporary client servers. Therefore, the Blockchain is used as a shared repository for this purpose.

6. Each of the servers represents a node of a private Blockchain. The Blockchain synchronizes data across all nodes, so data saved from one node will be available to the others.

7. To store and retrieve data from the Blockchain, a smart contract function is described, which is deployed on a private Blockchain network. They can be used to store and retrieve deployment details when needed.

8. At regular intervals, one of the temporary clients becomes a temporary server, and the previous master becomes a client. The new temporary server will redistribute all services again randomly.

The service distribution program and smart contracts run on different platforms. Therefore, the interface provided by Web3js is used to establish a connection between them. The main host is located on the first layer of the Blockchain in the first node, which is also temporary due to the property of changing position.

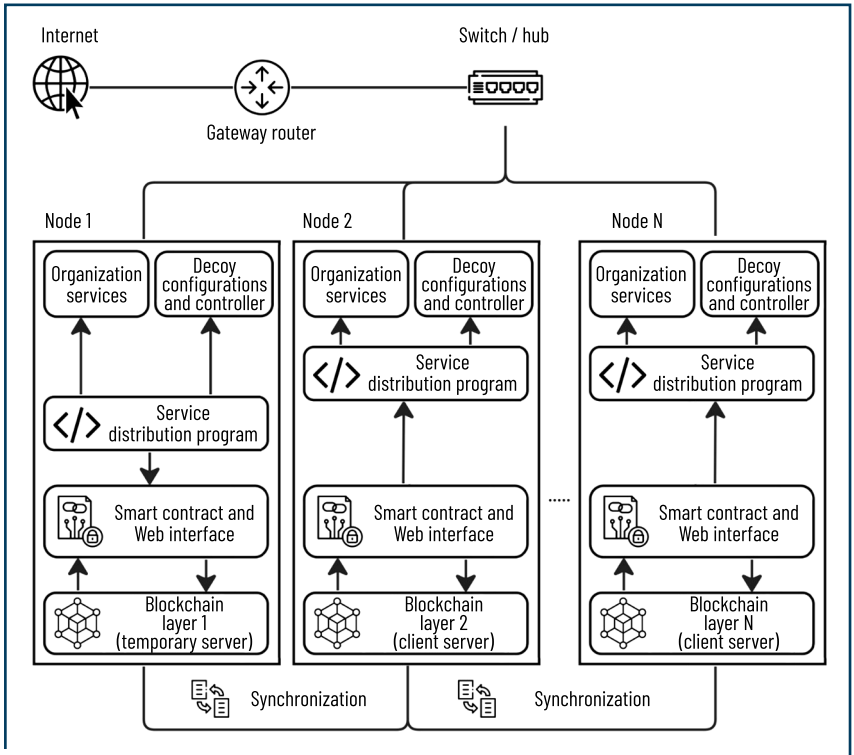


Fig. 4.5 Method of dynamic system built using software decoys

As a formal specification language, Alloy can be used to express a system based on the logic of the first layer [87]. The creation of a system overview, the theory of establishment, and the abstraction of atoms and relations are implemented in the Alloy system model.

Accordingly, the system behavior and some related notations are introduced in **Table 4.1**.

The system will perform a set of atomic actions, that is, actions = {generate, send, receive, wait, open, close, restore, compromise}. These actions and their parameters are given in Section 3.

Services on a host are abstracted by input/output events. The generate(data) and send(data, c_i) events represent the output data of the services, and the receive(data, c_i) event represents the receipt of data. From a security perspective, each host can operate in normal or compromised mode at the same time. Normal mode means that the host operates without malicious data and maintains normal operation. However, compromised mode indicates that the host operates maliciously and causes harm to itself. The states related to the running $host_i$ are divided into two categories: $Service_i^N$ for the normal mode and $Service_i^C$ for the compromised mode [88]. The worst case is that the host has broken down and stopped working in the broken mode $Service_i^B$. Thus, the states consist of three types {$service_n$, $service_c$, $service_b$}. The host will

work as the current mode until a transition $\rightarrow$ InterT occurs, which represents the transition relationship between the three different modes. The transitions shown in **Fig. 4.6** can be defined as follows:

$$Service_a \rightarrow T, InterTService_b : Service_a \in Service^N \wedge Service_b \in Service^C;$$
$$Service_b \rightarrow T, InterTService_a : Service_b \in Service^C \wedge Service_a \in Service^N;$$
$$Service_b \rightarrow T, InterTService_c : Service_b \in Service^C \wedge Service_c \in Service^B;$$
$$Service_c \rightarrow T, InterTService_a : Service_c \in Service^B \wedge Service_a \in Service^N.$$

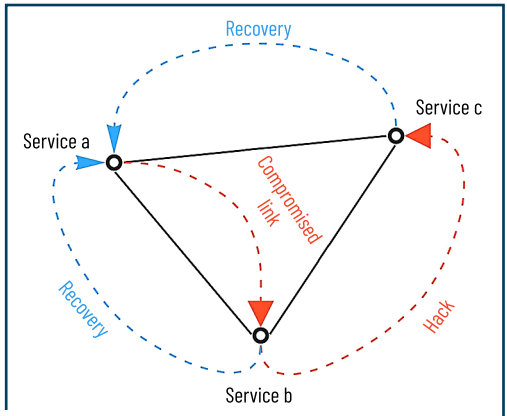


Fig. 4.6 Transitional states of links

A host consists of five parts $host_i = (id_i, Ports_i, Services_i, States_i \rightarrow T)$, where id_i – the host identifier, $Ports_i$ – a set of ports, $Services_i$ – a set of services, $States_i$ – a set of states and $\rightarrow T$ – a set of transition relations, given in **Table 4.1**.

Table 4.1 Transitional dependencies of the states of the links

Transition	Designation
$\rightarrow Table$	$\rightarrow IntraT \cup \rightarrow InterT$
$\rightarrow IntraT$	{normal state – normal state} {attacked state – attacked state} {compromised – compromised}
$\rightarrow InterT$	{attacked state – normal state} {compromised – normal state} {compromised – compromised state} {compromised – attacked state}

The basis of data communication in Blockchain technology is a consensus mechanism. Validator links check data coming from other links and decide by voting whether to allow information into the network. Any deviation (non-verification of data) compromises the link from which the request comes. Therefore, even internal attacks (if the attacker is in the network) are very difficult to carry out unnoticed.

Since the data transmitted between hosts ensures the normal operation of the system, it plays an important role in security analysis. It is assumed that each piece of data is generated by only one host [89]. The data may be malicious and contain some commands that lead to malicious activity. The host that generates the malicious data is considered compromised.

They are described as follows:

$$malicious(data) = \exists s^{generate(data)} \rightarrow Tables' : service \in Service^{C};$$
$$compromised(h_i) = \forall s \rightarrow Table, IntraTs' : service \in Service^{C} \wedge service' \in Service^{C}.$$

A host h_i with normal behavior is in the normal mode $normal(h_i) = compromised(h_i)$. It is assumed that if a normal host receives malicious data, it will switch to the compromised mode, further compromising itself. To simulate the propagation of unauthorized data during an attack, the following is obtained:

$$service_i^{compromise} \rightarrow Table, InterTService_i \exists Service_{i-1} : Service_{i-1} \rightarrow Table, IntraTservice_i.$$

To prevent a compromised host from intercepting the data transmitted during communication, the communication channel must be protected:

$$secure(c) = \forall (h_j, h_i) connectedState(h_j, h_i, c, States) \wedge \neg \exists s^{accept(data)};$$
$$\rightarrow InterTs' \wedge (compromised(h_i) \vee compromised(h_j).$$

The system consists of h hosts, as shown in **Table 4.1**. All states in these hosts illustrate the overall state of the system, i.e. $Service_{system} = Service_{h1} \cup Service_{h2} \cup ... \cup Service_{hh}$. The continuous authorized behavior of each host (e.g., sending data over a channel) ensures the normal functioning of the system. Any hosts communicate with each other by sending and receiving data over communication channels:

$$host_i, send(c, data) \rightarrow host_j, receive(c, data) hhi, send(c, data).$$

This indicates that sharing a single channel allows both connection and transmission of shared data. Data exchange can only occur when they are connected over a single communication channel. Therefore, let's define the following statement:

$$connected(host_i, host_j, c, States) = \exists host_j^{connect(c)} \rightarrow host_i \wedge host_i^{connect(c)} \rightarrow host_j.$$

During the communication process, the behavior of $host_i$ and $host_j$ is shown in **Fig. 4.7**. Host $host_i$ generates data and sends data to $host_j$ over a communication channel. After receiving data from $host_i$, $host_j$ can decide whether to accept or reject this data.

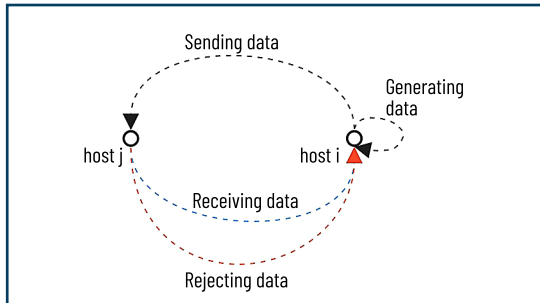


Fig. 4.7 Behavior during communication

Once $host_j$ receives and accepts malicious data, it becomes compromised:

$$host_i, generate(data);$$
$$host_i, send(c, data) \rightarrow host_j, receive(c, data);$$
$$host_j, accept(data) \vee host_j, discard(data);$$

If: $malicious(data) \wedge accept(data)$;
Then: $host_j$, *compromise;*
So: $compromisedState(h_j)$.

The normal host serves as a legitimate part of the system and ensures the normal functioning of its services for users. As mentioned above, {compromised – normal state} indicates that the compromised host becomes normal during the recovery action:

If: $compromisedState(host_j)$;
Then: $host_j$, *recover;*
So: $normalState(host_j)$.

The system abstracts and focuses on data transmission for subsequent analysis of attacks on security issues [90].

5 RESEARCH OF THE EFFECTIVENESS OF THE METHOD OF USING SOFTWARE DECOYS BUILT ON THE BASIS OF BLOCKCHAIN TECHNOLOGY AS ELEMENTS OF PROTECTION OF INFORMATION ACTIVITY OBJECTS

5.1 ANALYSIS OF THE SECURITY LEVEL OF THE SYSTEM USING SOFTWARE DECOYS BASED ON THE DEVELOPED MODEL AND SOLUTIONS FOR ITS PROTECTION

There have been many security issues in the past few years. Attacks are actions that use violence or something illegal to try to damage the security system of a network. Technologies to protect against all types of attacks ensure the stability of the system. This section mainly analyzes three types of attacks (i.e., sniffer attacks, scanning attacks, and DDoS attacks) and the corresponding solutions to solve these problems [130, 131].

First, these attacks are quite common in modern networks and are used by attackers to commit various types of cybercrimes. Sniffer attacks, scanning attacks, and DDoS attacks can be used to commit various types of cybercrimes, such as stealing confidential information, damaging websites, demanding ransom payments, and many more [132, 133].

Second, these attacks differ in their specificities and characteristics. For example, sniffer attacks involve intercepting traffic, which can lead to the disclosure of confidential information, scanning attacks involve finding vulnerabilities in the network and exploiting them by attackers to carry out attacks, and DDoS attacks involve attempting to stop a website from operating by overloading its servers [134, 135].

Third, the use of these attacks allows testing different security technologies and evaluating their effectiveness. For example, the effectiveness of protection technologies against sniffer attacks can be evaluated by testing data encryption methods, and the effectiveness of protection technologies against DDoS attacks can be evaluated by testing distributed request processing methods [136, 137].

5.1.1 DESCRIPTION OF THE EXPERIMENT: SNIFFER ATTACK

A sniffer attack is an illegal action by a third party to secretly obtain data transmitted between both sides of a communication. To demonstrate this type of attack, two hosts $host_0$ and $host_1$ are considered. Both are connected via a single communication channel $connectedState(host_0, host_1, c, States)$ and serve as normal hosts $normalState(host_0) \wedge normal(host_1)$ [134, 135]. In the sniffer attack scenario, there is a third host attack, and information is intercepted. When $host_0$ sends data to $host_1$, $host_{attack}$ secretly receives the transmitted data (i.e., $connectedState(host_0, host_{attack}, c, States)$), which should be sent to the only recipient $host_1$. Such an attack can be described as follows:

$$\left[host_0\right] generate(data);$$

$$\left[host_0\right] send(c, data);$$

$$\left[host_1\right] receive(c, data);$$

$$\left[host_{attack}\right] receive(c, data).$$

It is inevitable that communication data can be obtained by an illegal attacker. Let's try to prevent information disclosure. Therefore, to guarantee the security of the communication channel, data encryption should be performed [136]. To protect against sniffer attacks between $host_0$ and $host_1$, a secure c for the data is needed, that is, $connected(host_0, host_1, c, States)$ should be secured:

$$\forall : secure(c).$$

The developed system is encrypted with RSA 2048-bit encryption algorithm, and it cannot be decoded without the corresponding privacy key [137, 138]. Thus, the communication channel is protected in the proposed system and additionally prevents sniffer attacks [139].

5.1.2 DESCRIPTION OF THE EXPERIMENT: SCANNING ATTACK

A scanning attack is an action that sends requests to all ports of a target host in order to investigate the open ports and then use their errors to launch attacks. Attackers usually use a scanning tool to perform a scanning attack [140, 141]. To demonstrate the scanning attack, it is assumed that the host $host_2$ is scanned by $host_{attack}$. First of all, the attack sends a data request to all possible ports on the target host $host_2$. Second, the open ports will receive the data request and respond to the source host $host_{attack}$. Next, $host_{attack}$ receives these responses. Finally, $host_{attack}$ finds errors on $host_2$ and launches attacks according to the open ports. Such an attack can be described as follows [142–144]:

$$\left[host_{attack}\right] generate(request);$$

$$\left[host_{attack}\right] send(c, request);$$

$$\left[host_2\right]\left[port_i\right] receive(c, request);$$

$$\left[host_2\right]\left[port_i\right] generate(reply);$$

$$\left[host_{attack}\right] receive(c, reply);$$

$$\left[host_{attack}\right] compromise(host_2).$$

To eliminate the problems caused by a scanning attack, the scanned ports must be unpredictable to the attacker. To maintain the property of unpredictability, these ports are constantly changing. If an attacker scans an open port $port_i$ and intends to attack host $host_2$ using $normalState(host_2)$, the defender must close $port_i$ and open another $port_j$, which indicates that the scanned port information loses its value.

In fact, the service in $host_2$ has its own port number:

$$host_2 open(srv_i) = host_2, open(port_i);$$
$$host_2 close(srv_i) = host_2, close(port_i).$$

Therefore, to protect services from scanning attacks, all ports must be closed and opened periodically:

$$\forall p : close(port_i) \vee open(port_j).$$

In the designed system, there are four services that are periodically opened or closed on different hosts to protect against scanning attacks.

5.1.3 DESCRIPTION OF THE EXPERIMENT: DDOS ATTACK

A DDoS attack is an attack model to send a large number of requests to the target host. The host receives a temporary burst of requests, and a crash occurs. An illegal attack on a host generates many requests and sends them to the host $host_3$ using $normalState(host_3)$. After receiving these requests from $host_{attack}$, $host_3$ will wait for their responses. There will be no response from them, which indicates a waste of system resources and further consumption of $host_3$ resources until it is crashed. Such an attack can be described as:

$$\left[host_{attack}\right] generate(request1);$$
$$\left[host_{attack}\right] generate(request2);$$
$$\ldots$$
$$\left[host_{attack}\right] generate(requestn);$$
$$\left[host_{attack}\right] send(c, request1);$$
$$\left[host_{attack}\right] send(c, request2);$$
$$\ldots$$
$$\left[host_{attack}\right] send(c, requestn);$$
$$\left[host_3\right] receive(c, request1);$$
$$\left[host_3\right] receive(c, request2);$$
$$\ldots$$
$$\left[host_3\right] receive(c, requestn);$$
$$\left[host_3\right] send(c, reply1);$$

$$\left[host_3\right] wait(reply1);$$
$$\left[host_3\right] send(c, reply2);$$
$$\left[host_3\right] wait(reply2);$$
$$...$$
$$\left[host_3\right] send(c, replyn);$$
$$\left[host_3\right] wait(replyn);$$
$$\left[host_{attack}\right] breakdown(host_3).$$

The failure of $host_3$ leads to a single point of failure. To solve the problem, a distributed scheme should be considered. Compared with the traditional centralized host, a distributed system can cope with the single point of failure problem. A distributed system contains h hosts and $host \geq 2$. When $host_{attack}$ sends n requests, there are two possible situations for a distributed system:

1. DDoS attack on a single host. In this case, the host $host_3{:}breakdown(host_3)$ occurs. Even if $host_3$ cannot function, other hosts (i.e. $host_1$, $host_2$, $host_4$, ..., $host_h$ with $normalState(host_1) \wedge normalState(host_2) \wedge normalState(host_4) \wedge normalState(host_h)$ can still provide service to users and maintain the normal operation of the entire system, which can avoid a single point of failure.

2. DDoS attack on all hosts. In this case, n is taken as the maximum number of host crash requests, and each host shares the attack traffic. If $host_{attack}$ sends n requests, each host receives n/h requests. h distributed hosts greatly reduce the illegal traffic compared with a single host, which indicates that the hosts in the system are not affected by the failure.

If a traditional system encounters a DDoS attack, then:

$$host_{attack}^{DDoS^n} \rightarrow h.$$

If a distributed system encounters a DDoS attack, then:

$$host_{attack}^{DDoS^{n/h}} \rightarrow h_1;$$
$$host_{attack}^{DDoS^{n/h}} \rightarrow h_2;$$
$$host_{attack}^{DDoS^{n/h}} \rightarrow h_3;$$
$$...$$
$$host_{attack}^{DDoS^{n/h}} \rightarrow h_h.$$

The prototype system has five distributed and decentralized hosts to effectively mitigate the DDoS attack. In summary, three types of attacks have been analyzed and solutions to combat these attacks have been illustrated. It has been theoretically proven that the given scheme can protect against these attacks.

5.2 EXPERIMENTAL ANALYSIS OF THE SECURITY LEVEL DURING SOLUTION SIMULATION

Now it is necessary to evaluate the effectiveness of the developed dynamic distributed scheme for protecting software decoys. The implementation of the prototype system is carried out in Python, Java and Solidity (i.e. in the Blockchain programming language), the controller program interface is shown in **Fig. 5.1**. In addition, experiments are carried out on five personal computers (PCs) Windows 16 GB on which WM is installed and the Linux operating system (OS) is simulated with 8 GB of RAM to run services, one PC with Windows 32 GB on which WM is installed and the Linux OS with 16 GB of RAM is simulated to launch attacks of various scales and one PC with Windows with 32 GB of RAM for an authorized user. The services (MySQL v8.0.27, Apache v2.4.51, Vsftpd v3.0.5 and Nginx v1.24.4) and Solana v1.6.7 (i.e. the Blockchain platform used to form the private Blockchain) are installed on five server hosts. The total number of real services on different hosts is calculated to illustrate their average distributions. Three types of attack tests are performed: sniffer attack, scanning attack and DDoS attack.

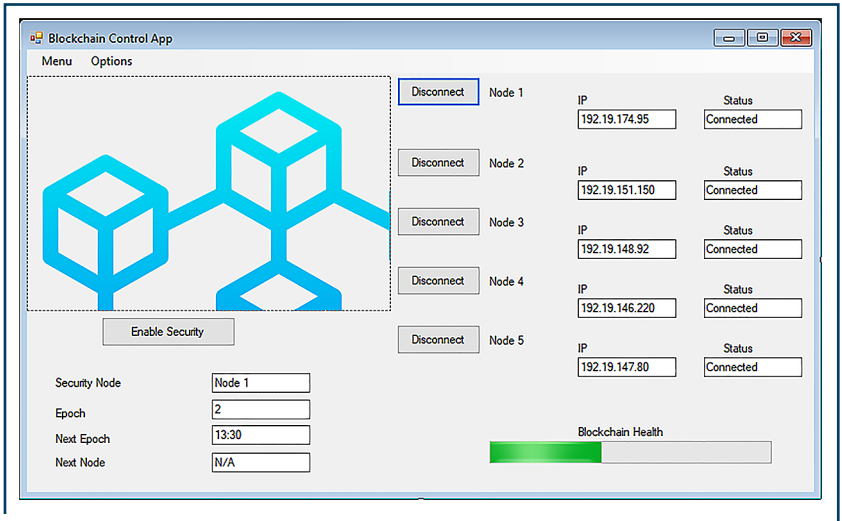


Fig. 5.1 Interface of the developed Blockchain controller program

Due to the transformation of services, attackers have no idea about the location of the real service. However, they can try to find a host with a large proportion of real services. Thus, the distribution algorithm is executed 60,000 times to test the distribution of real services. Since there are four types of services, there will be 240,000 real services in this experiment. The exact number of four services on five hosts is shown in **Fig. 5.2**. The proportion of different hosts is approximately 22 %, 20 %, 23 %, 19 % and 16 %, respectively. Overall, the distribution occurs with equal probability. All these percentages are above 16 %, indicating that real services are likely to be running on each host.

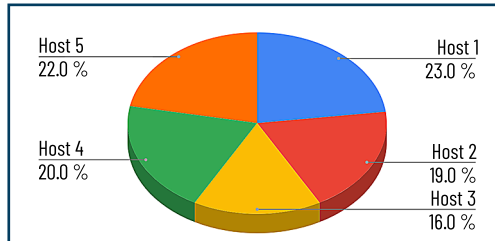


Fig. 5.2 Distribution of real services between hosts

5.2.1 STUDY OF THE PROTECTION OF THE SYSTEM USING SOFTWARE DECOYS BASED ON THE DEVELOPED MODEL AGAINST SNIFFER ATTACK

The sniffer attack is carried out through Wireshark v3.6.0. As shown in **Fig. 5.3**, the communication data between two servers is received secretly. As mentioned above, the communication information is encrypted using the 2048-bit RSA encryption algorithm [144, 145]. The sniffer attack is not performed on the real service, but on a fake one. The attack is taken over by the activated software decoy [146]. Even if the attacker receives the data transmitted during communication, it cannot obtain the plaintext from the encrypted text [147]. However, even if to assume that the attacker was still somehow able to decode the data, it will not receive the real information but the fake one, fed to it by the decoy protection system. In this case, the attacker will additionally have to spend a huge amount of time decoding, which illustrates that this scheme is effective for protecting against sniffer attacks [148–150].

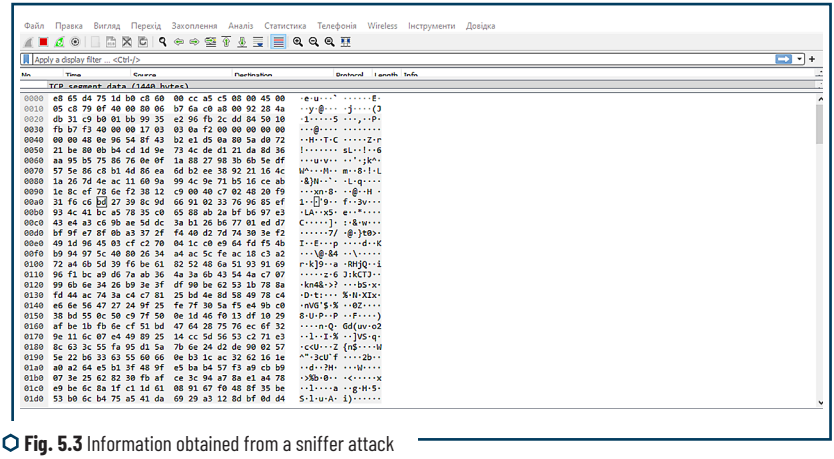


Fig. 5.3 Information obtained from a sniffer attack

5.2.2 STUDY OF THE PROTECTION OF THE SYSTEM USING SOFTWARE DECOYS BASED ON THE DEVELOPED MODEL AGAINST SCANNING ATTACK

Scanning is an effective step for attackers to find a system error. Usually, the attacker can obtain the IP address of the target in advance. In this experiment, the scanning attack is tested via Namp v7.92 (i.e., the scanning tool) [151, 152]. The five server hosts mentioned above are checked, as shown in **Fig. 5.4**. The scan command nmap -T4 -A -v 192.19.174.95 192.19.151.150 192.19.148.92 192.19.146.220 is used to scan these hosts to 192.19.146.220 and vice versa. Taking the IP "192.19.174.95" as shown in **Fig. 5.5**, for example, it turns out that ports 80 and 21 are open. They represent Nginx and Vsftpd respectively [153]. At this point, the attacker can start hacking the server. However, let's get the result that the ports are closed, as shown in **Fig. 5.5**. Since the detected services are closed, the attacker cannot obtain any service resources [154]. Even if the same port is open, the attacker is likely to access the decoy and obtain fake resources [155]. Due to the transformation property, the services are constantly changing. Even using a scanning tool, the attacker has no idea of the exact IP address of the service and falls into the decoy trap [156]. The decoy then logs the event in the attack log, allowing cybersecurity experts to assess the risks and learn about the attacker's actions [157]. A legitimate user does not need to scan ports. The very act of scanning is a breach of system security. Therefore, this scheme is effective in protecting against scanning attacks, which is advantageous in the comprehensive protection of computer networks built using the dynamic decoy method [158].

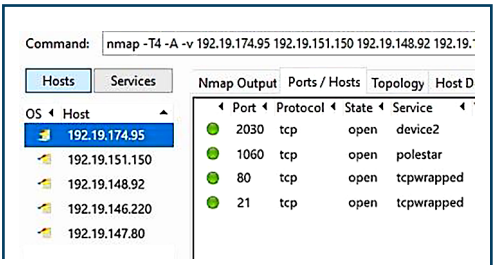


Fig. 5.4 First scan: result

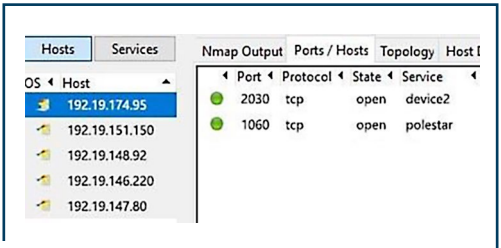


Fig. 5.5 Second scan: result

5.2.3 STUDY OF THE PROTECTION OF THE SYSTEM USING SOFTWARE DECOYS BASED ON THE DEVELOPED MODEL AGAINST DDOS ATTACK

As is known, DDoS attack is aimed at wasting PC resources, thus preventing the server from providing normal services and resources. The network performance and response time of static hosts and dynamic servers (i.e., the proposed scheme) during SYN DDoS attack are evaluated [159, 160]. The attack is tested by continuously sending SYN packets at different rates.

To evaluate the network performance, the SYN packet size for the attack is set to 73695 bytes in Hping3 v3.2.2, which indicates that the packet is divided into certain TCP packets. The network performance is measured using Iperf v3.10.1. **Fig. 5.6, 5.7** illustrate the impact of attack rate on network performance on effective throughput and TCP traffic. When the attack rate is 0 (i.e., no attacking packet), both types of hosts reach their maximum values of 736 (MB/s) and 100 (Mbps) in TCP throughput and TCP traffic, respectively [161]. However, as the attack rate increases, there is a sharp slowdown from 0 to 1000 packets per second. Apparently, the angle of incidence in static hosts is larger compared to the broken line of dynamic hosts [162]. As can be seen, there is a slow increase in the range from 1000 (packets/s) to 3000 (packets/s), and the value of dynamic hosts is still larger than that of static hosts. Thus, the dynamic decoy system has an advantage over static hosts in terms of network performance [163, 164].

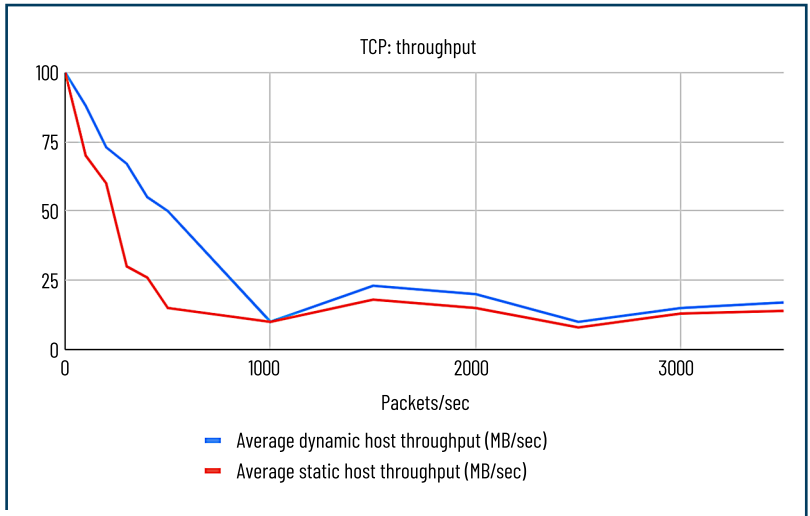


Fig. 5.6 TCP throughput comparison

To show the percentage relationship between the systems, it is possible to compare the average throughput of a dynamic host (*DH*) and a static host (*SH*). To compare the systems, let's calculate the

percentage relationship between the *DH* and *SH* throughputs using the formula: $\sum_n \left(\left(\frac{DH}{SH} - 1 \right) * 100 \right) / n$,

where *n* is number of attacks, *DH* is dynamic host through put during the attack, *SH* is static host throughput during the attack [165, 166].

Therefore, the average percentage value is 54 %. This means that a dynamic host (*DH*) is on average up to 54 % better at handling attacks than a static host (*SH*) [167, 168].

Therefore, the average percentage value is 204 %. This means that the dynamic host (*DH*) copes with data transfer during attacks on average by up to 204 % better than the static host (*SH*).

Conclusion: based on these data, it can be concluded that the dynamic host transmits data much more efficiently during attacks compared to the static host.

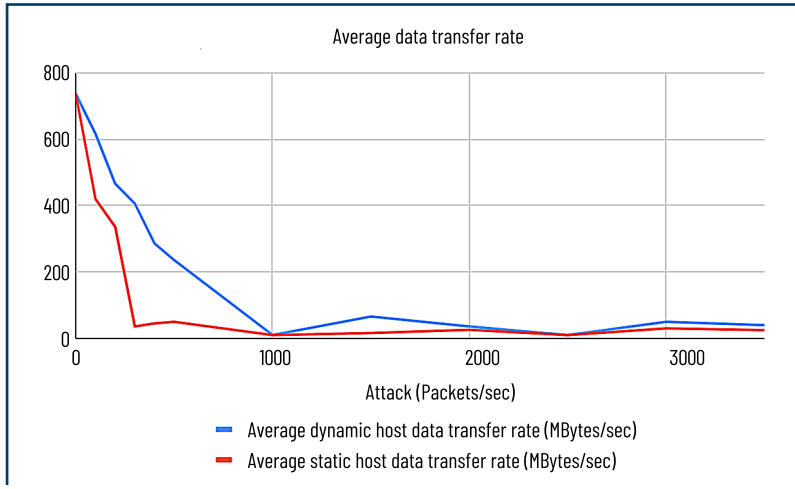


Fig. 5.7 Comparison of average throughput rate

5.2.3.1 ANALYSIS OF NETWORK RESPONSE TIME IN SYSTEMS USING THE DEVELOPED MODEL AND ITS ANALOGUES

Trafgen in netsniff-ng v0.6.7 is used to run the SYN attack test. Unlike the SYN packet mentioned in the network performance assessment, this type of packet consists of 64 bytes for the SYN flood attack. Since there are four types of services in the developed system, the average service response time becomes an indispensable evaluation indicator. The response time measurement is performed by Jmeter v5.4.2 for each service [169, 170].

The database query operator "select * from school" is used to measure the time it takes to retrieve the relevant data. As shown in **Fig. 5.8**, the static host does not respond with an attack rate of 14 (Kbps). However, the response time of the dynamic hosts seems to remain constant from 0 to 10 (Kbps) on the *X*-axis and reaches an infinite value after 60 (Kbps) on the *X*-axis [171]. The comparison with the dynamic hosts is striking, so the MySQL server of the static host suffers from a DDoS attack. Since the five distributed hosts distribute the attack load, the experimental curve of the dynamic hosts demonstrates their superiority in protecting against DDoS attacks [172, 173].

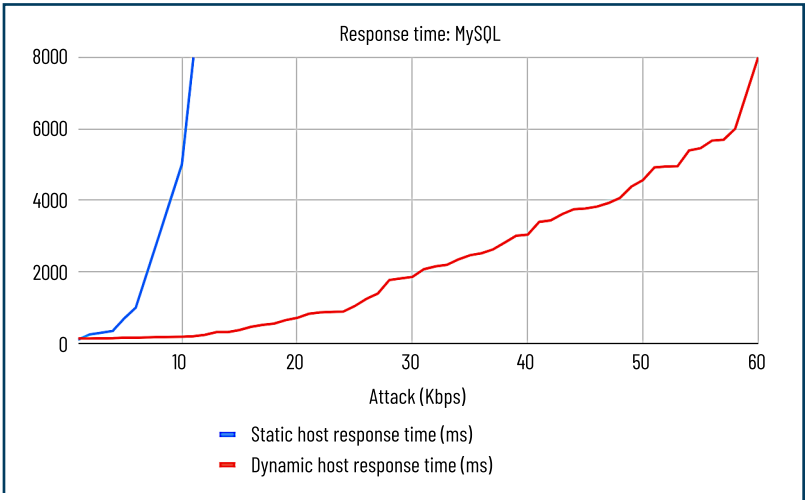


Fig. 5.8 Response time: MySQL

The average percentage is 34 %. This means that static hosts perform up to 34 % worse in response time during attacks than dynamic hosts.

Conclusion: based on this data, it is possible to conclude that dynamic hosts perform better in response time during attacks than static hosts. In particular, dynamic hosts perform up to 34 % better in response time during attacks on average [173, 174].

The loading time of the entire Apache web page is checked. In **Fig. 5.9** dynamic hosts take longer to load the web page than the static host. This is because the Blockchain mining operation exhausts some system resources, which becomes a key factor affecting the server response time. The response times of static and dynamic hosts are almost the same from 1 (Mbps) to 10 (Mbps), increasing slightly along the *X*-axis. In this case, both types of hosts are affected by the DDoS attack. The static server becomes unresponsive starting at 8.5 (Mbps), while the response time of the dynamic server is higher at the same attack rate, indicating that the dynamic Apache server can still respond even if the static server fails [170, 172].

So, the average percentage is 1 %. This means that static hosts are on average up to 1 % worse at responding to attacks than dynamic hosts.

Conclusion: based on these data, it is possible to conclude that dynamic hosts are better at responding to attacks than static hosts. The difference is 1 %, so in real scenarios there may be differences depending on the specific situation [172].

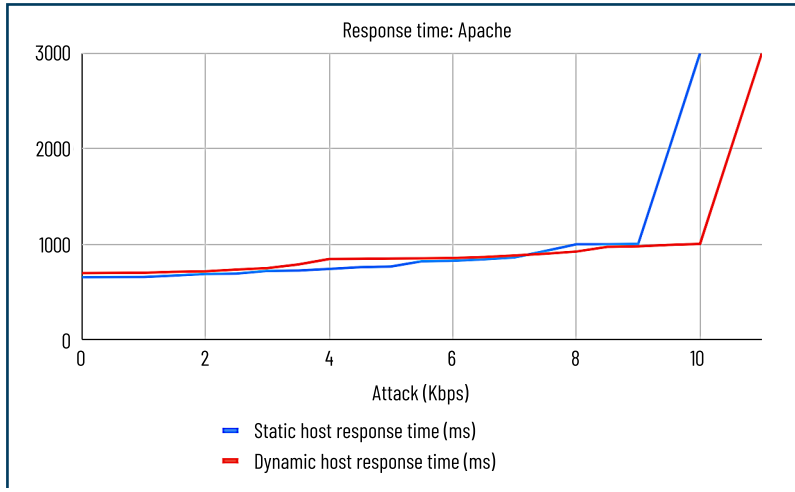


Fig. 5.9 Response time: Apache

The response time curves of Vsftpd and Nginx are shown in **Fig. 5.10, 5.11**. The response time of downloading a txt file from the Vsftpd server is measured during a DDoS attack. In **Fig. 5.10**, the response time of Vsftpd on the static host increases rapidly and reaches its infinity at 11 (Kbit/sec). Due to Blockchain mining, the overall trend from 0 to 10 (Kbit/sec) is slightly affected. However, the flat trend of the dynamic hosts curve indicates resilience to DDoS attacks. As mining operations on dynamic hosts exhaust system resources, Nginx is affected. As shown in **Fig. 5.11**, the average response time of Nginx on a static host outperforms that of dynamic hosts from 1 to 2.5 (Mbps) along the X-axis. After 2 (Mbps), the DDoS attack becomes the main factor affecting the response time. From 2 (Mbps) to 4 (Mbps), the dynamic hosts curve is always lower than the other, which means that the time on a static host is longer than that on dynamic hosts. It is known that Nginx creates symbols with less memory and high parallelism, so both curves maintain their smooth characteristics. However, the downtime of the static server occurs earlier than that of the dynamic one, which indicates the effectiveness of the designed scheme.

So, the average percentage is 13 %. This means that static hosts have an average of up to 13 % worse response time during attacks than dynamic hosts.

Conclusion: based on this data, it is possible to conclude that dynamic hosts have a better response time during attacks than static hosts.

Therefore, the average percentage is 16 %. This means that static hosts on average cope with response time during attacks by up to 16 % worse than dynamic hosts.

Conclusion: based on these data, it is possible to conclude that dynamic hosts cope with response time during attacks better than static hosts.

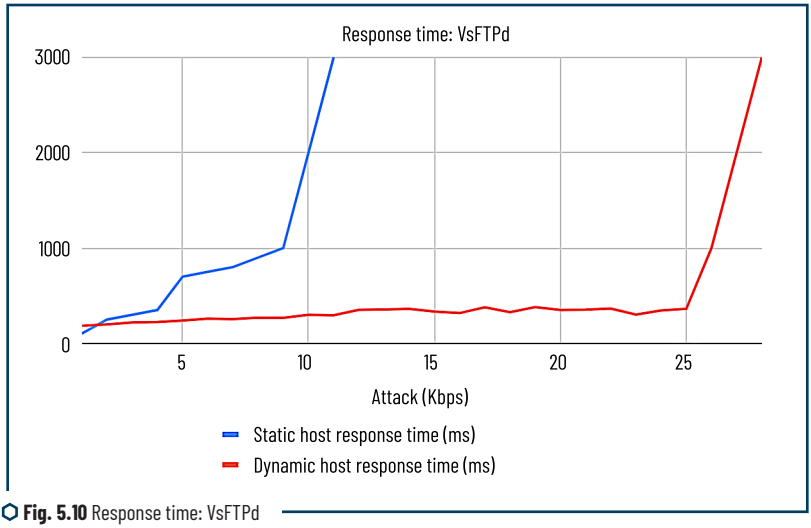


Fig. 5.10 Response time: VsFTPd

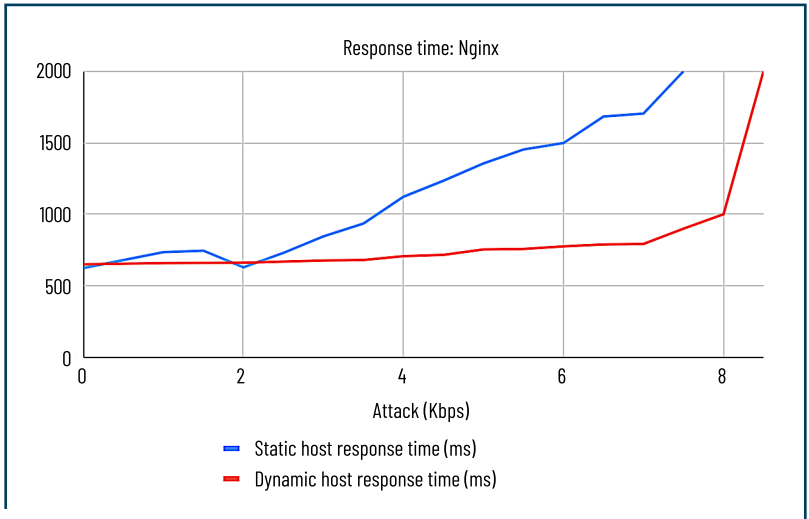
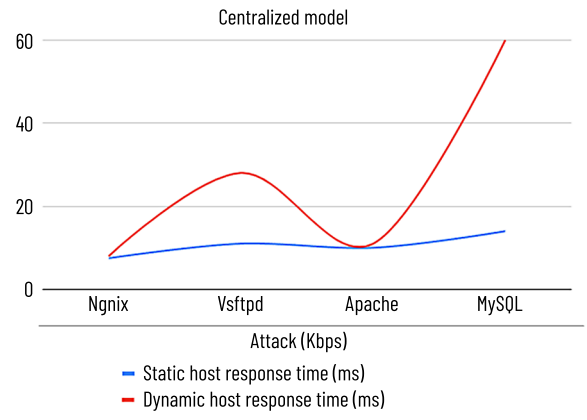


Fig. 5.11 Response time: Nginx

The same number of attacks were carried out on both the centralized system using software decoys and the dynamic system built on the developed method. The developed method of using software decoys, which is built on the use of Blockchain technology, requires more resources from the attacker to carry out an attack on the network: the power of computers, servers from which the attack is carried out, as well as more physical time, which increases the time for cybersecurity specialists to respond and counter the attack by up to 45 %. The biggest difference is not visible on all services: Apache and Nginx dynamic systems experience almost the same results as the central analogue during the attack. However, the Vsftpd and MySQL services require the use of significantly more resources from the attacker, which shows the effectiveness of the developed method in terms of protecting a computer network. A comparison of the effectiveness of the centralized model with the developed model of using software decoys during the attack is shown in **Fig. 5.12**.



⬡ **Fig. 5.12** Comparison of the effectiveness of the centralized model and the developed model during the attack

The total number of attacks is 60, 30 for the network with the developed model and the centralized one. One attack consisted of three different approaches that were used earlier and included DDoS attacks, sniffer and scanning. The attack is considered blocked if all three vulnerabilities were repelled. **Fig. 5.13** shows the result of the attacks for the centralized model. The programmed decoys in centralized models are not a protective mechanism, as mentioned earlier. They serve as a warning layer in the protective system and their task is to distract the attacker and collect and record information about the actions taken in the attack logs. Out of 30 attacks, only 4 were successfully blocked by all three parameters, which is 13 % of the total. This indicator tends to decrease with an increase in the number of attacks. It is also visible that 13 % of attacks passed completely by all parameters because the software decoy was found

and ignored and the attack continued immediately on the legitimate network. DDoS attack was blocked 11 times out of 30, which is 36 % of attacks of this type, scanning attack was blocked 17 times out of 30, which is 56 % of attacks of this type, sniffer attack was blocked 19 times out of 30, which is 63 % of attacks of this type.

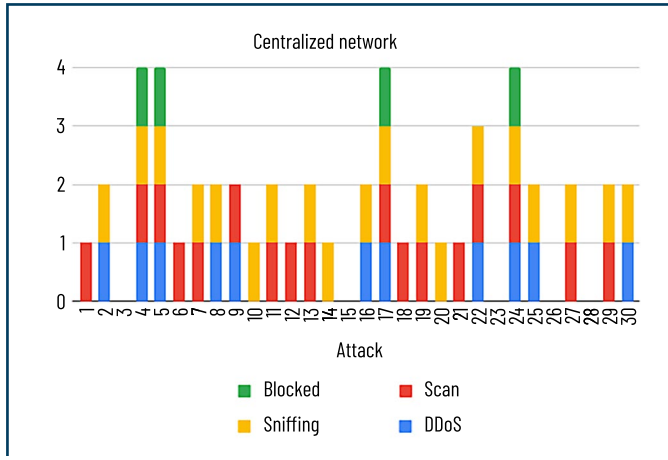


Fig. 5.13 The result of attacks on the centralized model

Fig. 5.14 shows the result of the attacks carried out for the developed model. Out of 30 attacks, half of them were successfully blocked, which is 50 % of the total number. Scanning and sniffer attacks could not be successfully carried out due to the translocation capabilities of the developed network, dynamic changes of hosts and constant exposure to software decoy. The weakest link in the network is DDoS attacks, the statistics of which affected the overall picture. Increasing the level of bandwidth and response time of services directly proportionally affects this result, since more computing power is required from the attacker to carry out the attack. Since the developed network consists of only five nodes (five personal computers), the load on them is accordingly greater. With an increase in the number of nodes in the network, the load will be more distributed and the results may improve several times. The DDoS attack was blocked 15 times out of 30, which is 50 % of attacks of this type, the scanning attack and the sniffer attack were blocked in all thirty attacks, which is 100 % of attacks of these types. However, from **Fig. 5.14** it is clear that software decoys built on the basis of Blockchain technology serve as a full-fledged protective mechanism and perform not only monitoring tasks but also tasks of direct protection of the computer network, which increases the overall level of network protection compared to analogues.

Comparing the centralized model with the decentralized one, it is possible to obtain the following results:

1. The protection of the decentralized network model during a DDoS attack is 14 % higher.
2. The protection of the decentralized network model during a scanning attack is 44 % higher. Almost twice.

3. The protection of the decentralized network model during a sniffer attack is 37 % higher.

4. The overall protection of a computer network built using programmed decoys based on Blockchain technology is 37 % higher compared to the centralized analogue, which is an increase in the global level of computer network protection by one and a half times.

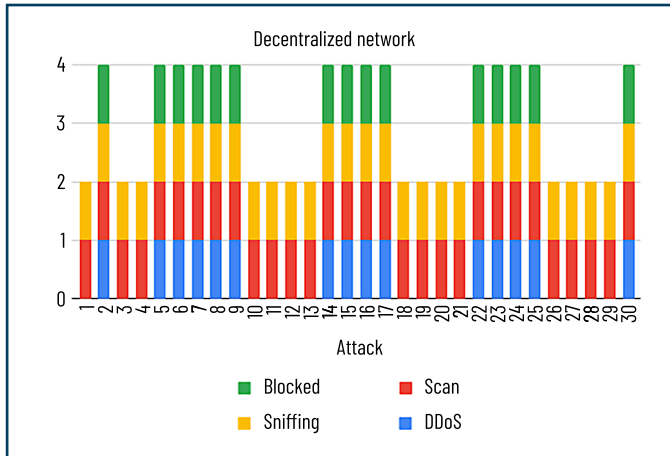


Fig. 5.14 The result of attacks on the developed model

5.3 COMPARATIVE ANALYSIS OF THE DEVELOPED DYNAMIC METHOD WITH STATIC ANALOGUES

To better illustrate the protective capabilities of the developed method, the response time of four services without any attack is summarized in **Table 5.1**.

Table 5.1 Service response rate when there is no attack

Service	Static host	Dynamic host
Ngnix	625 ms	650 ms
Vsftpd	103 ms	185 ms
Apache	653 ms	695 ms
MySQL	105 ms	135 ms

The times on a static host are always lower than on a dynamic host. Even if the complexity value in the genesis file is adjusted to "0x400" to reduce the computational overhead, the consensus mechanism

for block generation makes dynamic services less efficient in terms of response time. However, the final attack rates of dynamic hosts that lead to service failure show an advantage over static hosts, as shown in **Table 5.2**.

Table 5.2 Attack rate before the service stops accepting requests

Service	Static host	Dynamic host
Ngnix	7.5 MB/s	8.0 MB/s
Vsftpd	11 MB/s	28 kB/s
Apache	10 MB/s	11 MB/s
MySQL	14 MB/s	60 kB/s

Without attack traffic, the low efficiency of dynamic hosts leads to longer response times. However, with the increase of attack traffic on dynamic hosts, they show advantages in protecting the system, which indicates the effectiveness of the proposed scheme.

6 DEVELOPMENT AND RESEARCH OF THE EFFECTIVENESS OF THE CYBERCRIMES RESEARCH SYSTEM MODEL AT INFORMATION ACTIVITY OBJECTS

6.1 DEVELOPMENT OF A THREAT RESEARCH SYSTEM MODEL

One of the components of the cybercrime investigation system is the threat investigation system. The main task of this component in the cybercrime investigation system is to collect indicators of system compromise from threat sources, to detect potential or confirmed cyberattacks and cybercrimes [147, 149, 153].

The principle of operation of the threat investigation system can be described as follows. Let's denote the algorithm for collecting system compromise data as a function A, which runs on the threat investigation system, in which SL is the source list (SL for short) and TI is the threat investigation system (TI for short). ∑ indicates the final iteration process of each source in the list. *n is* the total number of sources. $Source_i$ represents the source from the list. $Verify(Source_i)$ represents the verification stage for each source (data format and API compatibility check) [169]. "*GetCred*" is the stage of obtaining credentials for each source [156]. "*TestCon*" checks the connection to the source using the obtained credentials [145]. *Integrate*() represents the integration of the source into the threat research system if the source is verified, the credentials are valid and the connection check is successful [173]. Therefore, the final formula of the proposed algorithm looks like this [164, 170]:

$$A(TI,SL)=\sum_{i=1}^{n} Integrate\left(Verify\left(Source_i\right),GetCred\left(Source_i\right),TestCon\left(Source_i\right)\right). \tag{6.1}$$

This algorithm ensures the secure integration of new threat intelligence sources into the threat intelligence system. It thoroughly verifies sources, processes credentials, verifies connections, and then integrates and schedules synchronization, while handling any errors that occur during the process [145, 164]. This systematic approach ensures that only trusted and compatible sources are added to the platform, enhancing its capabilities to monitor and respond to cyberthreats [173].

To verify the presence of an Indicator of Compromise in the threat intelligence system, the observed data is compared to an existing array of indicators of compromise (*IOCs*). If the *MatchScore* of *ObservedData* against *IOCs* is greater than or equal to a threshold value (*Threshold*): at least 1, then the presence of an indicator of compromise in the threat intelligence system is confirmed [169]. The formula for checking for indicators of compromise is as follows [149, 170]:

$$VerifiedIOC = TrueifMatchScore(IOCs, ObservedData) \geq Threshold, False. \tag{6.2}$$

This approach was implemented on the open source MISP threat intelligence system. The proposed threat intelligence system structure is shown in **Fig. 6.1**. This **Figure** shows the interaction of threat intelligence sources and cybercrime intelligence systems. The open source MISP system was identified as the

basis for the threat intelligence system. External threat sources that send indicators of compromise include IBM X-Force Exchange, AlienVault OTX, and CISA Automatic Distribution of Indicators of Compromise.

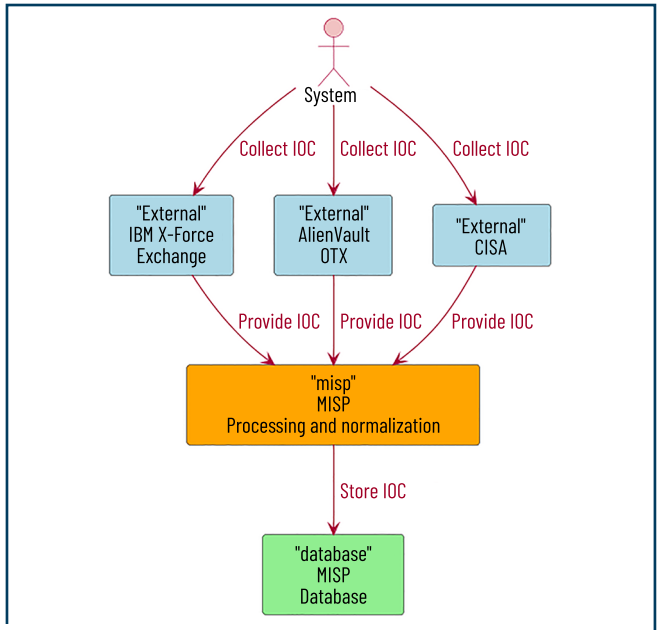


Fig. 6.1 Schematic representation of a threat research system

Deriving threat data starts by iterating over an array of threat sources. Each source contains important details such as name, URL, access key, data format, synchronization frequency, and filtering rules [145, 164]. For each source, the algorithm checks:

1. Is the source's data format one of the accepted formats? For example: STIX, TAXII, CSV, or JSON.

2. Is the source's API compatible with the system? That is, can it communicate effectively with the platform.

The algorithm then attempts to obtain the necessary credentials (e.g., API keys) for each source. It checks whether these credentials are valid and can successfully establish a connection to the source's URL.

If the source passes the check and the credentials are valid, the algorithm prepares to integrate the source into the MISP system. This involves compiling details such as the source name, URL, credentials, and data format. It then adds these details to the threat research system, effectively integrating the new source. After successfully adding the source, the algorithm sets a schedule for synchronization with the source. This synchronization occurs at a frequency specified in the source details, and includes special filtering rules to manage the incoming data.

6.2 VULNERABILITY MANAGEMENT COMPONENT

A vulnerability management system was developed to investigate vulnerabilities that can be used by an attacker to compromise an information system and to analyze the impact of an unpatched vulnerability on the occurrence of an information security event. This system is described by a set of the following functions [147, 164]:

1. Vulnerability detection function:

$$VD(S) = \sum_{i=1}^{n} v_i(S), \tag{6.3}$$

where $VD(S)$ – the system vulnerability detection function, (S) represents a single vulnerability detected in the system, and n is the total number of vulnerabilities detected in the information system.

2. Vulnerability assessment function:

$$VS(v_i) = (\text{CVSS v3.0}). \tag{6.4}$$

For vulnerability assessment, it is proposed to use the Common Vulnerability Scoring System (CVSS v3.0), which is a standardized system for assessing the severity of security system vulnerabilities. In this case, $VS(v_i)$ is a vulnerability assessment based on the standardized CVSS v3.0 value determined by the scanning system. However, CVSS v3.0 does not take into account individual characteristics of the information system that may affect the severity of the vulnerability and the priority of the fix, in particular the phase of system development, the presence of security measures, and the criticality of data. Therefore, to take into account these factors, it is proposed to introduce a weighting factor that can be applied by organizations as needed to improve the assessment of research into cybercrimes caused by unpatched vulnerabilities. Based on expert judgment, a score was determined for each of the weighting criteria. The results of surveys of 20 information security experts were collected and the scores are proposed in **Tables 6.1–6.3**. This weighting factor is not mandatory, but it allows information security analysts to take into account the individual characteristics of the system under study. Taking into account the weighting factor, the overall vulnerability assessment of the system is described by the following formula:

$$VA(S) = \sum_{i=1}^{n} VS(v_i) \times W(P, SC, IC). \tag{6.5}$$

This formula assumes that $VS(v_i)$ is CVSS v3.0 and the weighting factor is based on the following criteria: system development phase (P), availability of security measures (SC) – a total factor calculated by multiplying the factors of each of the security measures and the information classification (IC) processed by the system. The justification for these criteria and the corresponding factors are proposed in **Tables 6.1–6.3**. The weighting factor determines the impact of these factors on the overall vulnerability assessment of the system, but it is worth noting that these estimates should be adjusted in accordance with a certain risk

tolerance and organizational policy. The weight formula, taking into account the above factors, can be presented as the average value of these three factors:

$$W = \frac{P + SC + IC}{3}. \tag{6.6}$$

Table 6.1 Criterion: information system development phase (*P*)

Development Phase	Coefficient	Rationale
In active use	1.0	Information systems in active use typically have the highest priority because vulnerabilities in these systems can directly impact the organization. A breach or failure here typically has a significant impact on the information system and the organization
In development	0.75	A system in development is less critical than one in active use, but it may be a replica of a system in use or contain confidential information and intellectual property of the organization
Proof of concept (POC)	0.25	Proof of Concept (POC) has the lowest impact because it is often isolated and used for experimental purposes. The risk and impact of vulnerabilities here are typically lower compared to other environments

Table 6.2 Criterion: security measures (*SC*)

Availability of security measures (*SC*)	Coefficient	Rationale
Firewalls and network segmentation	0.8 (measures implemented), 0.9 (partially implemented), 1.0 (measures not available)	Firewalls and network segmentation significantly reduce the risk of exploiting vulnerabilities
Intrusion detection and prevention systems (IDS and IPS)	0.7 (measures implemented), 0.85 (partially implemented), 1.0 (measures not available)	Effective IDS and IPS can detect and potentially prevent exploits
Endpoint protection (antivirus, EDR, etc.)	0.75 (measures implemented), 0.9 (partially implemented), 1.0 (measures not available)	Endpoint protection can reduce the risk of an attacker exploiting a vulnerability if there are behavioral malware signatures or exploits
Access control and authentication	0.8 (measures implemented), 0.9 (partially implemented), 1.0 (measures not available)	Access controls and authentication mechanisms limit the possibility of unauthorized access
Data encryption (at rest and in transit)	0.85 (measures implemented), 0.95 (partially implemented), 1.0 (measures not available)	End-to-end encryption protects data even if other layers of security are compromised
Patch management	0.8 (measures implemented), 0.9 (partially implemented), 1.0 (measures not available)	Timely detection and response to patches will make it impossible to exploit them
Information security management and security event management (SIEM)	0.75 (measures implemented), 0.9 (partially implemented), 1.0 (measures not available)	Using basic SIEM capabilities with real-time analysis makes it possible to detect and respond to attempts to exploit vulnerabilities

Table 6.3 Criterion: information classification (*IC*)

Information classification	Coefficient	Rationale
Confidential information	1.0	Systems that handle sensitive data receive the highest score, as a breach could result in significant legal, financial, and reputational damage to an organization
Public information	0.5	Systems that handle public information are considered less risky because the data is already publicly available and a breach would not result in the disclosure of sensitive information. However, manipulation of this type of data can lead to the spread of false information, especially when the information is used in the context of operational public information or news

These formulas can be implemented in the cybercrime investigation system, namely in the process of assessing vulnerabilities in information systems. The process involves identifying individual vulnerabilities, evaluating them according to various criteria, and then summarizing these scores to obtain an overall score.

Taking into account the results of the research conducted in the previous sections, a vulnerability management system was developed that uses the following open source solutions: DAST-type scanning solutions – OWASP ZAP, SCA – OWASP Dependency Check, OpenVAS: tools provide a comprehensive approach to vulnerability management, covering a wide range of security assessments from the infrastructure to the code level. The vulnerability management system is schematically presented in **Fig. 6.2**.

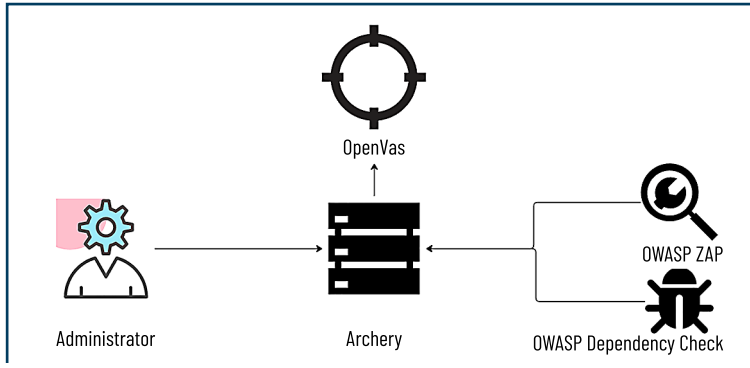


Fig. 6.2 Schematic representation of the vulnerability management system

The integration of these tools into the vulnerability management component makes it possible to ensure thorough vulnerability detection and vulnerability management at different levels of information systems.

OpenVAS offers extensive infrastructure scanning capabilities and, as an open source tool, is cost-effective and benefits from community-driven updates. OWASP Dependency-Check: specializes in detecting vulnerabilities in project dependencies. It can be integrated into CI/CD pipelines, making it a tool for

detecting vulnerabilities early in the software development lifecycle. OWASP ZAP (Zed Attack Proxy): provides dynamic application security testing required to detect vulnerabilities while the information system is running. Its proxy interceptor capabilities allow for deep analysis of application behavior, offering both active and passive scanning options. Archery Security Tool: acts as a central platform for aggregating and managing vulnerabilities discovered by other tools. This simplifies tracking, analyzing, and reporting on security flaws, making it easier to prioritize and effectively remediate vulnerabilities.

6.3 DEVELOPMENT OF A METHODOLOGY FOR INVESTIGATING CYBERCRIMES BASED ON ANOMALY DETECTION USING THE ISOLATION FOREST MODEL AND GPT, TAKING INTO ACCOUNT INFORMATION SYSTEM VULNERABILITIES AND THREAT INTELLIGENCE DATA

This section proposes a methodology for investigating cybercrimes for information system infrastructure components based on the Isolation Forest anomaly detection algorithms and GPT models, taking into account information system vulnerabilities and threat intelligence data. The model of the cybercrime investigation system for information system infrastructure components is presented in **Fig. 6.3**.

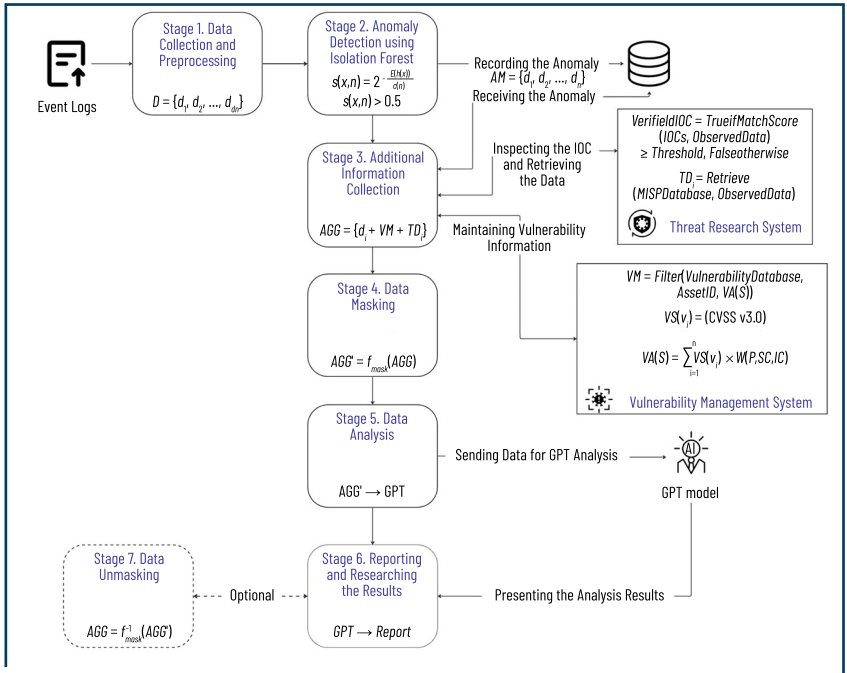


Fig. 6.3 Cybercrime investigation system model for information systems infrastructure components

It is worth noting that this cybercrime investigation system model is not dependent on the GPT model version and is built taking into account the principle of continuous improvement, therefore it allows information security teams to integrate any external systems with GPT model support.

The developed model offers cybercrime investigation according to the following methodology:

Stage 1. Data collection and preprocessing. Event logs from components of different levels of the information system infrastructure are collected centrally. These can be network traffic logs, system access logs, transaction records, etc. It is important to add that the data needs to be cleaned and pre-processed to ensure that they have a suitable format for analysis. To mathematically describe an array of logs for the model, especially in the context of cybercrime investigation using methods such as Isolation Forest, it is necessary to represent the logs in a structured data format, usually as a matrix. Each row in the matrix represents a separate log record, and each column represents a separate characteristic or attribute extracted from the logs. Let D (from English Data) be a matrix representing an array of logs. Suppose there are n log records and m characteristics extracted from each log record.

Then, D is represented as a matrix:

$$D = \{d_1, d_2, ..., d_{dn}\}. \tag{6.7}$$

Stage 2. Anomaly detection using Isolation Forest. Training the Isolation Forest model on a dataset with normal behavior is the first step. Isolation Forest is effective at detecting anomalies, after which it isolates the anomalies. To train the model, it is necessary to extract relevant records from the logs that may indicate anomalies. This may involve techniques such as PCA (Principal Component Analysis) to reduce dimensionality or more complex feature development. The process of training Isolation Forest involves building multiple isolated trees. The mathematical description of the model training process can be described as follows:

1. Initialization: let the forest have T trees. For each tree t_i, where (i = 1,2,3, ..., T), a random set of samples (S_i) is selected from the array D.

2. Recursive partition in tree t_i: a random function f_j of a random partition value v between the minimum and maximum values of f_j in the random set of samples S_i is selected. It is divided into two subsets based on the partition value in the function f_j. The partitioning continues until all samples are isolated or a certain depth limit is set.

3. Path length calculation: the path length for sample $h(x)$ in each tree t_i. Anomaly score for each sample in next step.

4. $$s(x,n) = 2 - \frac{E(h(x))}{c(n)}, \tag{6.8}$$

where $s(x,n)$ – the anomaly score for sample x, $E(h(x))$ – the average path length in the tree, and $c(n)$ – the average path length in a random binary search tree.

5. Processing anomalies detected by the isolation forest. For each anomaly, a threshold value for the anomaly score is determined, for a normal event this value is from 0 to 0.5. The value $s(x,n) > 0.5$ indicates

the presence of an anomaly, and a score very close to 1 indicates the exact presence of an anomaly. If such an anomaly is detected, the anomalies are added to the set of anomalies *AM*:

$$AM = \{d_1, d_2, ..., d_n\}, \tag{6.9}$$

where *AM* – the array of anomalies, *d* – the detected anomaly, and *n* – the number of anomalies.

Stage 3. Additional information collection about the information system and its components. It is proposed to use additional data sources to add context to the anomalies. This information should be obtained from the threat research system and the vulnerability analysis system. If the event log with the anomaly contains an indicator of system compromise from the specified threat sources or a vulnerability detected by vulnerability detection systems, this data should be collected into a single data array. To collect data from the threat research system, it is proposed to check for the presence of compromise indicators in the threat research system:

$$VerifiedIOC = TrueifMatch(IOCs, ObservedData) \geq Threshold, False.$$

And perform the following function to collect compromise indicators, if the received value is > 1.

$$TD_i = Retrieve(MISPFDatabase, ObservedData), \tag{6.11}$$

where TD_i represents a function for retrieving threat data, which may include indicators of compromise (IOC), *i* – a sequence number. Retrieve is an operation that represents the process of requesting information from the MISP system. *MISPDatabase* is the MISP database. *ObservedData* is a confirmed indicator of compromise. The next step is to collect data on vulnerabilities in the information system infrastructure. This step is described by the following formula:

$$VM = Filter(VulnerabilityDatabase, AssetID, VA(S)), \tag{6.12}$$

where *VA*(*S*) – the overall vulnerability assessment of the system, *AssetID* – the asset number, and *VulnerabilityDatabase* – the vulnerability database of the Archery system.

$$AGG = d_i + VM + TD_i, \tag{6.13}$$

where *AGG* – an array of anomalies, d_i – the detected anomaly, *i* – a sequence number.

Stage 4. Data masking. Before feeding the log data (*AGG* matrix) into the cybercrime research model, a data masking technique must be applied. Data masking can be represented as a function f_{mask} that takes the original data matrix *YM* and returns the masked matrix:

$$AGG' = f_{mask}(AGG). \tag{6.14}$$

Stage 5. Data analysis. The masked matrix, threat source data, and information about the vulnerabilities of the information system are sent to the GPT model for information synthesis:

$$AGG' \rightarrow GPT. \tag{6.15}$$

Stage 6. Reporting and researching the results. At this stage, the research of the results obtained by the GPT model is carried out:

$$GPT \rightarrow Report. \tag{6.16}$$

Stage 7 (Optional). Data unmasking. In cases where the raw data needs to be obtained for detailed analysis or reporting, it is necessary to use the inverse function of the masking function:

$$AGG = f_{mask}^{-1}(AGG'). \tag{6.17}$$

The actions that a cybercrime research model should perform according to a specific methodology are presented in **Fig. 6.4**.

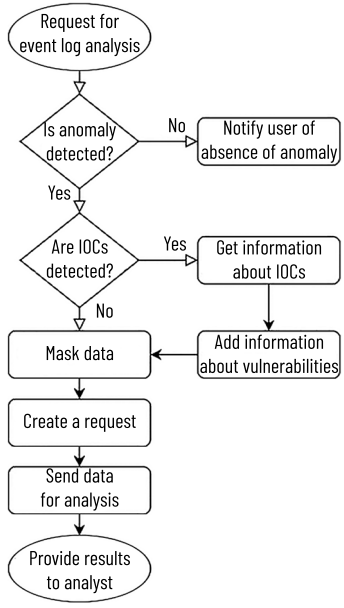


Fig. 6.4 Flowchart of actions

When the analyst provides the event logs to the cybercrime investigation model collected using a dynamic decoy system based on Blockchain technology, the model activates the anomaly detection algorithm. The trained model detects the anomaly and automatically collects it into a single data set.

If an anomaly is detected, the model queries the vulnerability analysis system and receives information about the vulnerabilities of third-party libraries and scans for vulnerabilities of the application and infrastructure of the information system.

The next check is to check the following indicators of system compromise, in particular IP addresses and domain names in the threat investigation system.

If indicators of compromise are present, the model adds information about the threats to the data set for analysis.

The next step is to query the vulnerability data, if a vulnerability is detected, the vulnerability data is collected by the model into a separate data set, which the model will use for analysis using GPT.

The generated data array is masked, all IP addresses and domain names are replaced with pseudo-random ones.

The masked data array is sent to the GPT model for analysis.

After analysis, the data is unmasked and the results of the verification are analyzed by an information security analyst.

The following section describes in detail the practical implementation of the cybercrime investigation system model and the principle of its operation.

6.4 PRACTICAL IMPLEMENTATION OF THE CYBERCRIME INVESTIGATION SYSTEM MODEL

The notations used to model the system are described in **Table 6.4**, and **Table 6.5** describes the system actions.

Algorithm 1 has been developed to describe the operation of the cybercrime investigation system.

Algorithm 1. Cybercrime investigation.

The get(log) function receives an array of event logs $\{log_1, log_2, \ldots, log_n\}$ in .log format. The information obtained is processed by the Isolation Forest model to detect anomalies. When an anomaly is detected, the system records anomalies using the record (anomaly) function in the database:

$$Isolation\ Forest(analyse(\text{logs})) \rightarrow record(anomaly);$$
$$\rightarrow \{Arr_1, Arr_2, \ldots, Arr_n\}\ else\ inform(anomaly\ not\ detected).$$

Next, the check (vulnerabilities) function requests information from the vulnerability detection system about the presence of vulnerable components of the information system:

$$if\ check(vulnerabilities) = True \rightarrow then\ Record_{I}n_{A}rr : V_1, V_2, \ldots, V_n\ \ else \rightarrow Noaction.$$

The check (IOC) function checks for the presence of indicators of system compromise in the threat research system:

$$if\ check\left(IOC\right) = True \rightarrow then\ Record_in_Arr : \left\{IOC_1, IOC_2, ..., IOC_n\right\} else \rightarrow Noaction.$$

The form(array) function forms a data array for analysis by the GPT system and the formed data array is masked by the mask(array) function:

$$form\ request\left(array\right) \rightarrow mask\left(array\right) \rightarrow array'.$$

The formed data array is sent to the GPT-based research system using the connection API using the send (array) function:

$$send\left(array'\right) \rightarrow GPT.$$

Table 6.4 Notations used to model the system

Name	Designation
Logs	$\{log_1, log_2, ..., log_n\}$
Vulnerabilities	$\{V_1, V_2, ..., V_n\}$
(IOC)	$\{IOC_1, IOC_2, ..., IOC_n\}$
IP	$\{IP_1, IP_2, ..., IP_n\}$
Domains	$\{d_1, d_2, ..., d_n\}$
Anomaly	$\{A_1, A_2, ..., A_n\}$
Array	$\{Arr_1, Arr_2, ..., Arr_n\}$
Result	$\{R_1, R_2, ..., R_n\}$

Table 6.5 Description of system actions

Function	Description
1	**2**
get (logs)	The Isolation Forest model receives data for analysis by the algorithm
analyse (logs)	The event logs are analyzed by the Isolation Forest algorithm to detect anomalies
record (anomaly, array)	The system writes anomalies to the data array, if detected
check (vulnerabilities)	The vulnerability management system is checked for vulnerabilities
record (vulnerability)	The system writes vulnerabilities to the data array, if detected
check (IOC)	The threat analysis system is checked for indicators of compromise

Continuation of Table 6.5

1	2
record (IOC)	The system writes the indicator and threat to the data array, if detected
mask (array)	The data masking function replaces IP addresses and domain names with pseudo-random ones
form request(array)	The system generates a data array for analysis
send (array)	The system sends the data array for analysis to a GPT-enabled system
receive (result)	The system receives the analysis result from the GPT-enabled system
demask (array)	The data unmasking function changes pseudo-random IP addresses and domain names to real ones
inform (result)	The system sends the analysis results to the user
inform (anomaly not detected)	The system notifies the user that there is no anomaly

The analysis results are received by the receive(result) function, in the next stage, the data is unmasked by the demask(array) function and transferred to the analyst by the inform(result) function. In this way, the system processes the data and provides a clue to the analyst about the possible root cause of the incident or cybercrime. A UML sequence diagram was created for the described system, shown in **Fig. 6.5**. The diagram outlines the interactions and processes in the system, involving the analyst, the cybercrime investigation system, and the external GPT model. This diagram provides a visual representation of the sequence of operations, including log reception, investigation, vulnerability and threat scanning, data masking, connections to external systems, and the final report of the results.

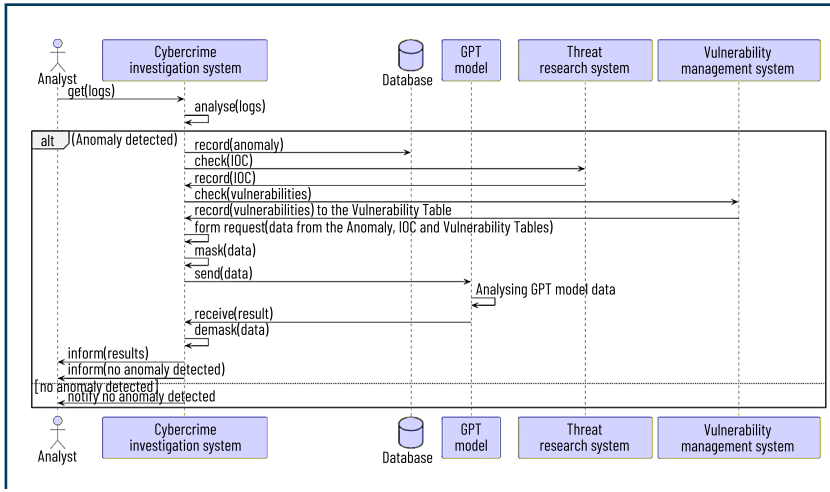


Fig. 6.5 Sequence diagram

The software architecture of the model is presented using the C&C view in **Fig. 6.6**, which indicates the elements of the system, namely its services and connections between the system services. It is worth noting that the microservice type of architecture was chosen for the system, since the system implemented based on this type of architecture is fault-tolerant, provides the ability to scale and flexible choice of technologies. These characteristics became an important factor in choosing the type of system architecture, since microservices allow easy scaling of individual program components, since each service can be scaled independently based on demand and each microservice can be created using different technologies (programming languages, databases, etc.), which allows using different technologies for the needs of each specific service, for example: vulnerability analysis, collection of indicators of compromise or GPT model. Therefore, the cybercrime investigation system was implemented using services in a containerized environment.

The system components are described in **Table 6.6**.

Table 6.6 System components

Component	Description
Service: Analyzer	The analyzer service is implemented in the Python programming language and is the main service of the model. This service performs the following functions: collecting information about the anomaly from the service, system vulnerabilities from the Archery service and indicators of system compromise from the MISP service. This service generates a request to the GPT model. Sends data for masking and sends the masked data to the GPT model. Upon receiving the result, the analyst reports the results of the study, or the absence of an anomaly
Service: Isolation Forest Model	The service is implemented based on the Isolation Forest model in the Python programming language, is a pre-trained model and receives data for analysis. The result of execution is a detected and documented anomaly
Service: Masker	The service is implemented in the Python programming language, which masks the data generated for analysis and unmasks the data received from the GPT model
Service: Hashicorp Vault	The Hashicorp Vault service is used to store API keys
Service: Arhery	The Archery service aggregates data from vulnerability detection systems and is an implementation of the described vulnerability management component
Service: MISP	The MISP service aggregates data from threat sources and is an implementation of the described threat research system
Service: GTP Service	The external GPT model is integrated into the cybercrime research system
Database: MySql	A database implemented on MySql. The database records information about detected vulnerabilities, anomalies, and indicators of system compromise

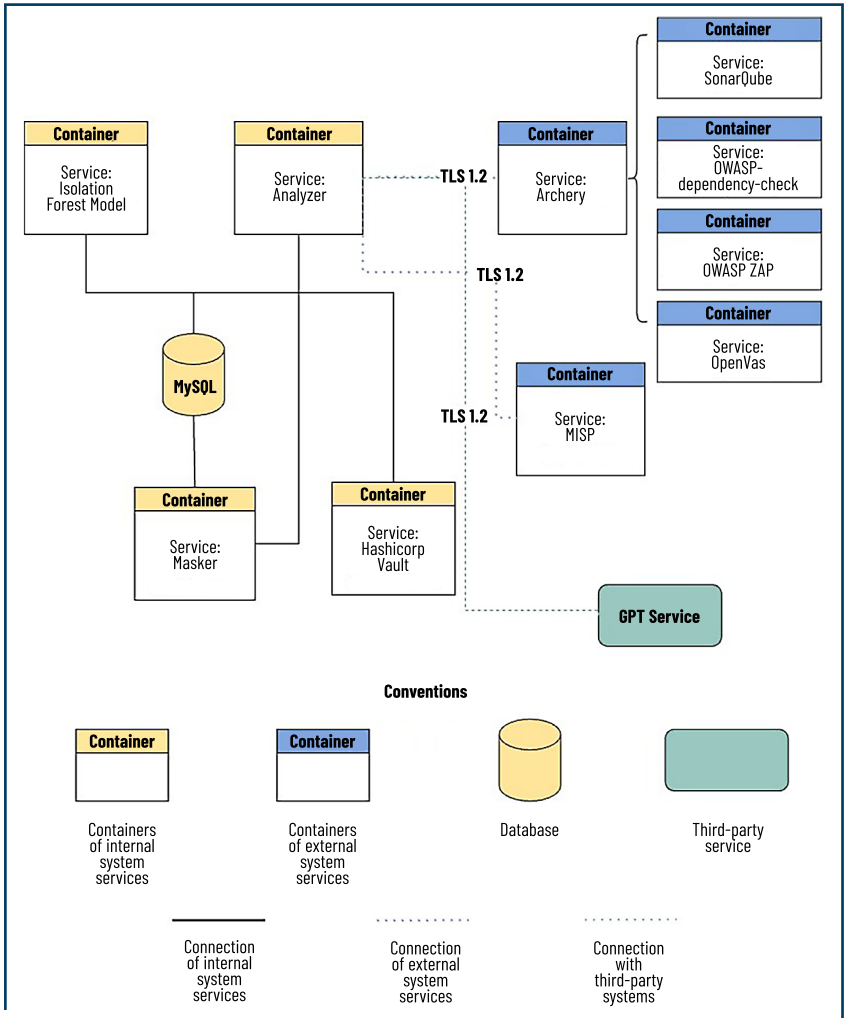


Fig. 6.6 Component diagram (C&C)

6.5 PREPARING A VULNERABLE ENVIRONMENT FOR TESTING A CYBERCRIME INVESTIGATION SYSTEM MODEL

At the current stage of information systems development, cybersecurity is mainly focused on protection that detects and prevents cyberattacks. However, it is much more important to regularly check the

security status of the organization to strengthen cybersecurity protection, as the IT environment becomes more complex and competitive [71].

This section presents the results of a simulation of cyberattacks on a prepared vulnerable environment, the simulation results are provided to the cybercrime investigation system for analysis. The environment for generating event logs consists of a decoy system based on blockchain technology, the main nodes of which are Nginx services, which are used as reverse proxy services and containers for the vulnerable OWASP JuiceShop application. This environment is schematically presented in **Fig. 6.7**.

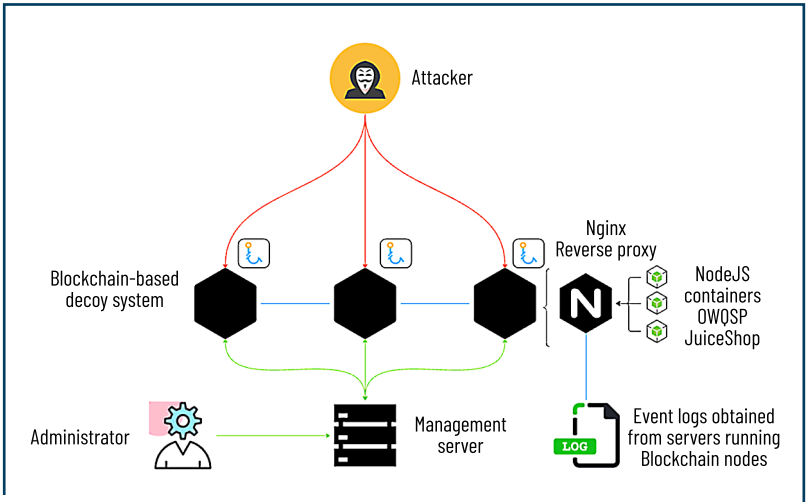


Fig. 6.7 Attack simulation environment

For experimental analysis, the following cyberattacks were simulated: an attack using automated vulnerability scanning and exploitation tools, a Directory Traversal attack, injections, and attempts to disrupt the program logic.

Using tools such as Nessus to scan for vulnerabilities is a measure to identify and eliminate weaknesses in a network or system. These tools are designed to detect a wide range of vulnerabilities, ranging from software flaws to incorrect configurations. The main advantage of using these tools is their comprehensive nature, they provide detailed information about potential security gaps. However, attackers can also use these solutions to identify vulnerabilities for further attacks on a selected system. Detecting this type of attack and preventing it in a timely manner can prevent further information security incidents.

Injection attack simulations, such as SQL injection and XSS, offer a real-time assessment of how well an organization's systems can withstand common and sophisticated attack methods. These simulations mimic the actions of cybercriminals, providing a realistic test of security measures. The advantage here is that it allows to evaluate the effectiveness of security protocols and identify specific areas where protection

may need to be strengthened. This approach to security testing helps you fine-tune your defenses against common attack vectors.

A Directory Traversal attack, also known as Path Traversal, is aimed at accessing files and directories that are stored outside the root folder of a website. By manipulating variables that refer to files using dot-dot-slash (../) and similar constructs, an attacker can move up the directory tree from the root of the web server to access arbitrary files or directories.

Detecting users attempting to break into application logic is critical to detecting insider threats or compromised accounts. The advantage of this approach is its emphasis on the human element of cybersecurity. By monitoring anomalies in user behavior, such as unusual login times or unexpected data access, organizations can quickly identify and investigate potential security incidents. This method is particularly effective at detecting threats that may be missed by automated systems, including subtle actions that may indicate hacking or malicious insider activity. The following sections describe the experiments conducted, the results of which were recorded in event logs for further analysis by the cybercrime investigation system.

6.5.1 TRAINING THE ISOLATED FOREST MODEL

Due to the use of Nginx event logs to successfully detect an anomaly during the experiment, it is necessary to train the Isolated Forest model on NGINX logs. Training involves several stages, starting from understanding and preprocessing the data to training the model. The test_nginx.log file containing 50210 requests, including anomalies and normal behavior, was used to train the model.

The Nginx event logs contained information such as remote IP addresses, timestamps, HTTP request methods, response status codes, and user agents. Each of these fields can potentially help in detecting anomalies (for example, unusual access patterns or error levels). Since the Isolated Forest only works with numeric data formats, the information from the event logs was obtained and converted to numeric data format for the model to work according to the following principles:

1. IP address: converted to numeric format.

2. Timestamp: converted to a datetime object and numeric functions defined.

3. HTTP method: obtained from the query string.

4. Requested URL: to analyze the URL, it was broken down into components - protocol, domain, path, and query string. The urllib Python library was used for this. The following numeric characteristics were defined for the URL:

– URL length: abnormally long URLs may indicate suspicious activity, such as SQL injection or path traversal attacks, and are not typical of normal user activity;

– path depth: the query depth was calculated by counting the number of slashes in the URL path. For each application, a maximum query depth can be defined, a value greater than this depth is anomalous;

– special characters: the presence of certain special characters (for example, "%", ".." or unusual encoding) may indicate attempts to exploit vulnerabilities.

5. HTTP Version: retrieved from the query string.

6. Status Code: indicates the result of the query (e.g. 404 – not found, 200 – successful).

7. Response size: the size of the response in bytes.

8. User agent: this field was defined as a bot characteristic using a binary flag (0 or 1) indicating whether the user agent is a bot.

The data prepared for the model was loaded into the model_training.cvs file. For which the attributes of anomalous behavior were defined. An Isolation Forest model was initialized on the trained data. Isolation Forest assigned an anomaly score to each observation from the trained data. This score was used to determine whether the observation was an anomaly. The final step was to save the model for later use, which was provided by using the joblib library.

To summarize, training the Isolation Forest model on Nginx logs involved converting the raw log data into a format that the algorithm could process, training the model to detect anomalies, and then interpreting those anomalies in the context of web server traffic and activity.

6.5.2 COLLECTION OF ADDITIONAL INFORMATION

Collection of additional information occurs from the vulnerability management system and the threat research system. Previously, a vulnerability scan was performed for the JuiceShop application.

Infrastructure scanning was performed using the OpenVas application, one low-level vulnerability was detected during the scan. Static scanning was implemented using OWASP Dependency Check, which detected 12 critical vulnerabilities, 26 high-level vulnerabilities, and 29 medium-level vulnerabilities in third-party libraries used by the information system. Dynamic scanning was performed using the OWASP ZAP application, which detected 2 medium-level and 5 low-level vulnerabilities. Scans were uploaded to the Archery vulnerability management system. The vulnerability scan results are presented in **Fig. 6.8**.

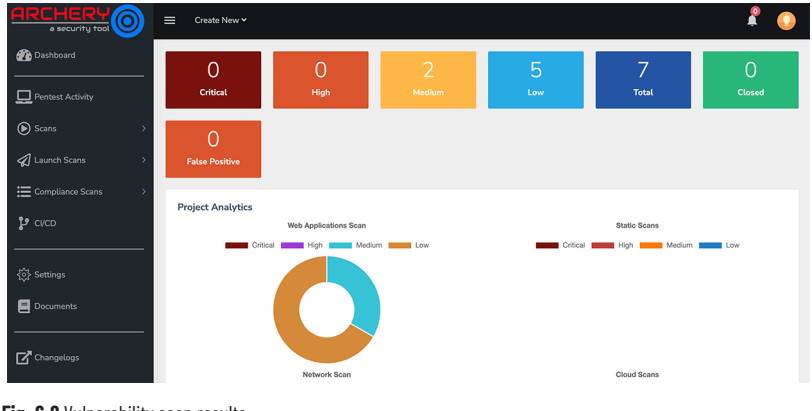


Fig. 6.8 Vulnerability scan results

All of the above-mentioned scanning systems are implemented taking into account the principle of microservice architecture and are implemented in the implementation of the cybercrime investigation system using Docker containers to ensure free scaling of the system. Secure communication between systems is provided using the TLS 1.2 algorithm. The integration of the vulnerability management system with the cybercrime investigation system is carried out using the connection API. To securely store the credentials of the service user used for the connection API, the Hashicorp Vault key store was implemented.

For the detected anomaly, an algorithm was implemented to obtain IP addresses from NGINX logs and check them in the threat investigation system implemented on the MISP Docker container. It is assumed that the IP addresses in the logs correspond to the standard IPv4 format with the use of a regular expression to extract them. This regular expression may need to be adjusted if the logs have a different IP address format. Interaction with the MISP API uses the basic structure of an HTTP POST request. API requests are configured according to MISP requirements. In addition, the service user credentials are stored in the Hashicorp Vault key store.

6.6 EXPERIMENTAL ATTACK EXECUTION

This section describes in detail the stages of experimental attacks on the system under study. It presents scanning attacks, injection attacks, Directory Traversal and program logic violation attacks.

6.6.1 EXPERIMENTAL ATTACK EXECUTION USING AUTOMATED SCANNING TOOLS

Vulnerability scanning was performed using Nessus. This application is a traditional method used to detect security vulnerabilities in network systems, but it also performs a number of automated attacks. An attack using automated scanning tools is schematically presented in **Fig. 6.9**.

An experiment of an attack using automated scanning tools includes:

1. Scan initiation: the scanning system Scanner begins the scanning process by identifying open ports and services. This is represented as the Scanner initiating the Scan function.

2. Target selection: the Scanner selects a target host, the Target, to assess for vulnerabilities. This is described as the Scanner selecting the Target.

3. Sending scan requests: the Scanner sends a series of requests to the Target. This action is denoted as the Send Scan Request function. These requests are intended to verify various aspects of the Target's security.

4. Target host scanned: the Target receives the scan requests.

5. Response analysis: the Scanner analyzes the responses from the Target to identify potential vulnerabilities.

6. Vulnerability report: the Scanner generates a report with a detailed description of the vulnerabilities and potential risks found, presented.

The main purpose of scanning with Nessus solutions is to identify security weaknesses in the target system. This attack can be used by an attacker to further exploit the identified vulnerabilities. The result of

scanning the tested system based on OWASP JuiceShop is the resulting Nessus scan report. The results are recorded in the Nginx event logs for further analysis.

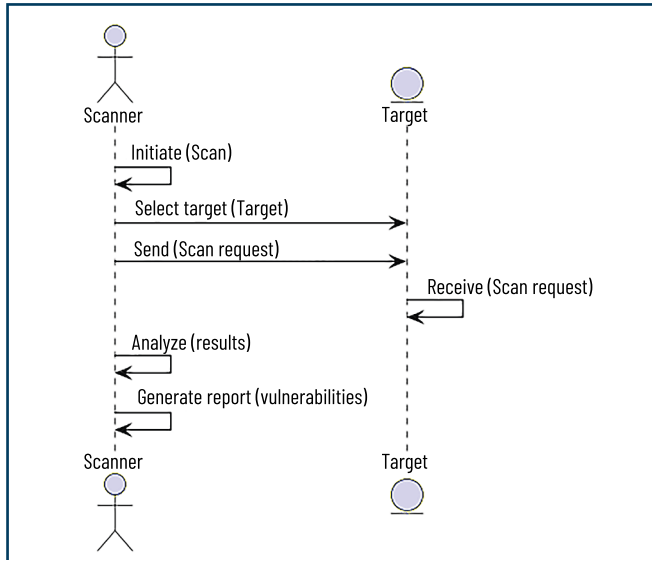


Fig. 6.9 Attack using automated scanning tools

6.6.2 EXPERIMENTAL INJECTION ATTACK

Injection attacks, such as SQL injection, are malicious methods used by attackers to exploit vulnerabilities in applications by injecting unauthorized code into queries or commands. The main purpose of injection attacks is to gain unauthorized access or perform unauthorized actions on the target application. The BurpSuite tester program was used to test injection attacks. The injection scanning attack is schematically presented in **Fig. 6.10**.

For the experiment, the attack is deployed in the following sequence:

1. The Tester program starts the process. In the diagram, this is represented as the Tester program performing the initiate (injection) function.

2. Creating a malicious request: the Tester creates a malicious request that is designed to exploit vulnerabilities in the target program.

3. Sending a malicious request to the target: the created malicious request is sent to the target program.

4. The program processes the input data: the target processes the received input data. Unaware of the malicious actions of the request.

5. Executing malicious code: the malicious input causes malicious actions by the target, such as unauthorized access to data or its modification.

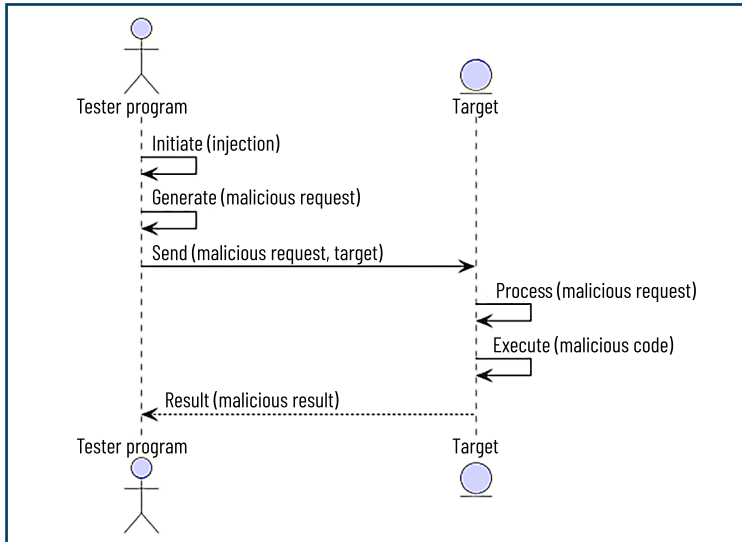


Fig. 6.10 Injection attack

The main goal of an SQL attack is to obtain the definition of the database schema and attempt to log in to the system with administrator rights. An XSS attack is implemented to introduce third-party content into the information system.

6.6.3 EXPERIMENTAL IMPLEMENTATION OF DIRECTORY TRAVERSAL ATTACK

The goal of the Directory Traversal attack is to access files and directories that are stored outside the intended accessible directories of the web server. This type of attack exploits vulnerabilities in the web application (or web server) that cannot properly sanitize user-entered file paths. The dirbuster application is used to simulate this type of attack. A schematic experimental implementation of the Directory Traversal attack is presented in **Fig. 6.11**.

For the experiment, the attack is deployed in the following sequence:

1. The Tester program starts the process. In the diagram, this is represented as the Tester program executing a malicious request that contains directory traversal sequences (for example, '../').

2. Sending the malicious request: the Tester program sends a request to the target server. The server processes the request: the target server processes the request without properly sanitizing the input data.

3. Jumping to a restricted directory: the server gains access to a directory or file outside the intended directory as a result of Directory Traversal.

4. Compromise of data: the Tester gains access to confidential files or directories, which can potentially lead to data leakage.

The results of the experiment implementation are recorded in the event logs.

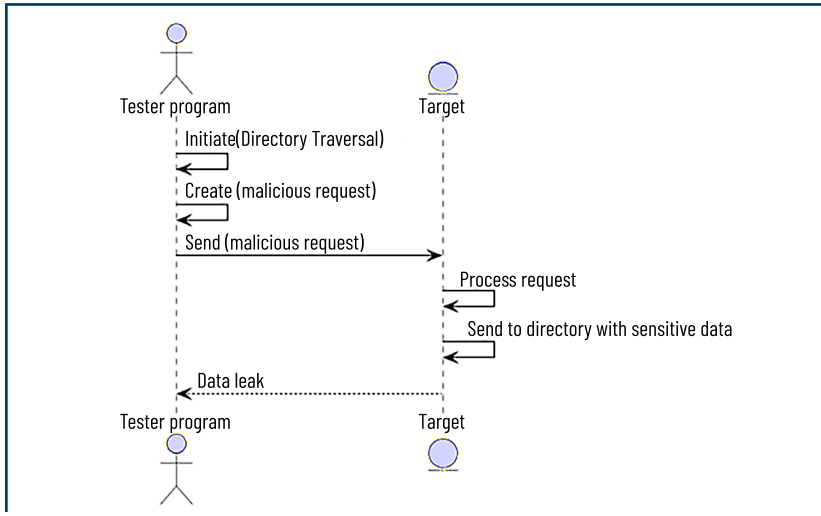


Fig. 6.11 Directory Traversal Attack

6.6.4 EXPERIMENTAL ATTEMPT TO VIOLATE THE LOGIC OF THE PROGRAM

This experiment simulates user actions caused by violation of the logic of the program. This includes entering data that the program is not intended to process, using logical flaws to attempt to disrupt the normal operation of the program. Schematically, the experimental attempt to violate the logic of the program is shown in **Fig. 6.12**.

For the experiment, the attack is deployed in the following sequence:

1. The user starts by entering a request that uses known logical flaws in the program.
2. The program processes the received request.
3. The program does not detect an anomaly and the logic of the work is violated.
4. In response to the violation, the program processes the request and returns the result of the request.

In the case of the testing application, the Burp Suite tester program is used, with which an attempt to reduce the price of the product by the user is simulated. The results are recorded in the Nginx event logs for further analysis.

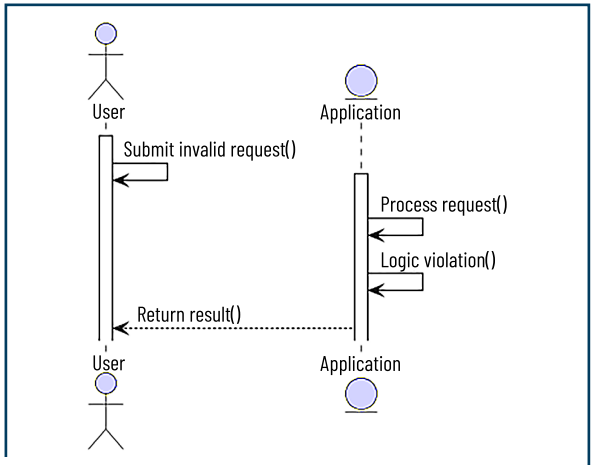


Fig. 6.12 Attempt to violate the program logic

6.7 ANALYSIS OF THE RESULTS

This section presents a thorough analysis of the operation of the proposed model of the cybercrime investigation system for the components of the information systems infrastructure. It describes an analysis of the effectiveness of the cybercrime investigation system using the proposed model. Also, the results obtained are compared with the open source Wazuh event investigation and monitoring system and the analysis performed by information security specialists.

To study the event logs, the traditional SIEM system Wazuh was used and two information security analysts were involved, for which the time of event detection by the SIEM system Wazuh and the total time of investigation of cybercrimes by information security analysts were measured. Time measurement is performed using Google Stopwatch.

6.7.1 ANALYSIS OF THE SPEED OF DETECTION AND INVESTIGATION OF ATTACKS USING AUTOMATIC SCANNING TOOLS

For this study, Nessus Community Edition was used, which simulated a scanning attack with a standard set of rules. For the experiment, event log records generated during automatic vulnerability scanning using Nessus Community Edition were used and 10 experiments were conducted with 10 data sets containing 100 identical event log records, each of which contains at least one anomaly. The study was conducted every hour for 10 hours. As shown in **Fig. 6.13**, the average anomaly detection time determined during these experiments is 5.1 seconds by the comprehensive cybercrime investigation system and the average

information security event detection time by the Wazuh SIEM system is 13.1 seconds. Therefore, the cybercrime investigation system is up to 61 % faster than the Wazuh SIEM system for detecting cybercrimes. The comparison of anomaly detection times is presented in **Fig. 6.13**.

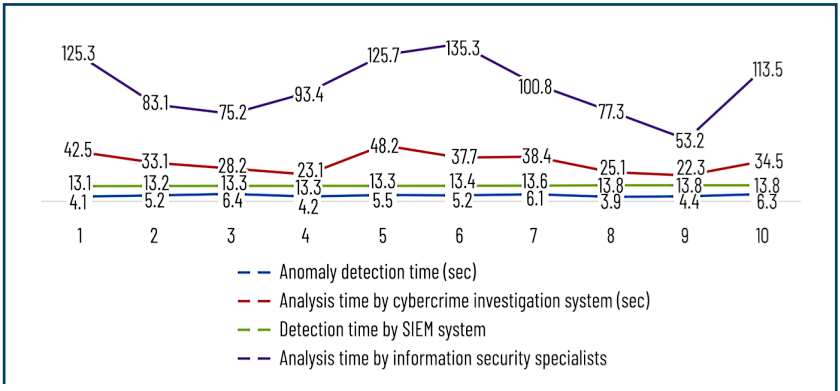


Fig. 6.13 Comparison of scan attack analysis time by a comprehensive cybercrime investigation system and a traditional approach

The next indicator taken into account is the cybercrime analysis time. The average time for cybercrime analysis by information security specialists using Wazuh SIEM data (this time does not take into account reporting) is 98.28 seconds and 33.31 seconds, respectively, by the cybercrime investigation system (this time takes into account receiving a detailed report). It has been experimentally confirmed that the cybercrime investigation system can increase the speed of cybercrime investigation by approximately 66 %, based on the criterion of cybercrime investigation speed.

Analyzing the results obtained, it was found that the cybercrime investigation system detects anomalies for components of the information system infrastructure up to 66 % faster, which can focus analysts' attention on a potential attack. Also, the cybercrime investigation system can increase the efficiency of cybercrime investigation by approximately 61 %.

6.7.2 ANALYSIS OF THE DETECTION SPEED AND INVESTIGATION OF INJECTION ATTACKS

To measure the speed, event logs generated during the execution of the injection attack were used and 5 experiments were conducted. During each experiment, 5 event logs were used, each of which contained 100 records. The study was conducted for 5 hours, one experiment every hour. The experiments determined that the time spent on anomaly detection was 5.66 seconds for the comprehensive cybercrime investigation system and 11.02 seconds for the Wazuh SIEM system. Therefore, it was found that the cybercrime investigation system is up to 48 % faster than the Wazuh SIEM system for detecting cybercrimes.

The average time for cybercrime investigation by information security specialists using the Wazuh SIEM system data was 98.52 seconds and 19.14 seconds, respectively, for the cybercrime investigation system. In this case, the implementation of a comprehensive cybercrime investigation system using GPT with an analysis time of 19.14 seconds can reduce the time for event log analysis by approximately 80 % in percentage terms compared to the SIEM system. The results of the study are presented in **Fig. 6.14**.

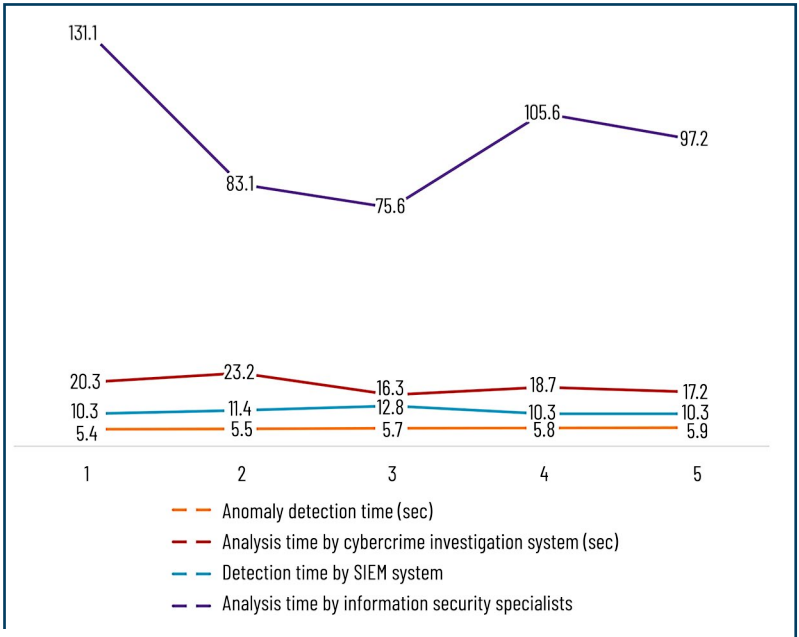


Fig. 6.14 Comparison of injection attack analysis time by a comprehensive cybercrime investigation system and a traditional approach

The results of the conducted experiment determine that the cybercrime investigation system detects information security events faster and can reduce the time for examining event logs by up to 80 % in percentage terms compared to the SIEM system without reducing efficiency.

6.7.3 ANALYSIS OF THE SPEED OF DETECTION AND INVESTIGATION OF DIRECTORY TRAVERSAL ATTACKS

To measure the speed, event logs generated during Directory Traversal attacks using Dirbuster were used and 5 experiments were conducted. During each experiment, 5 event logs were used, each containing 100 records. The study was conducted for 5 hours, one experiment every hour. The average anomaly detection time is 6.68 seconds for the comprehensive cybercrime investigation system and the time for detecting

an information security event by the SIEM system is 7.44 seconds. Therefore, the cybercrime investigation system can reduce the event detection time by up to 7.80 % in percentage terms. The lowest anomaly detection time is 6.1 seconds for the cybercrime investigation system and 7.4 seconds for the Wazuh SIEM system. Therefore, as is seen, the cybercrime investigation system is up to 17 % faster than the SIEM system for cybercrime detection, if to consider the detection time as a performance criterion. The next indicator defined for evaluating the performance is the cybercrime analysis time. The average time for cybercrime analysis by information security specialists using SIEM system data is 23.56 seconds and 14.54 seconds, respectively, for the cybercrime investigation system. In this case, implementing a comprehensive cybercrime investigation system using GPT with an analysis time of 14.54 seconds can reduce the event log analysis time by up to 38 % in percentage terms compared to the SIEM system. **Fig. 6.15** shows a comparison of the analysis time of a Directory Traversal attack by a comprehensive cybercrime investigation system and a traditional approach.

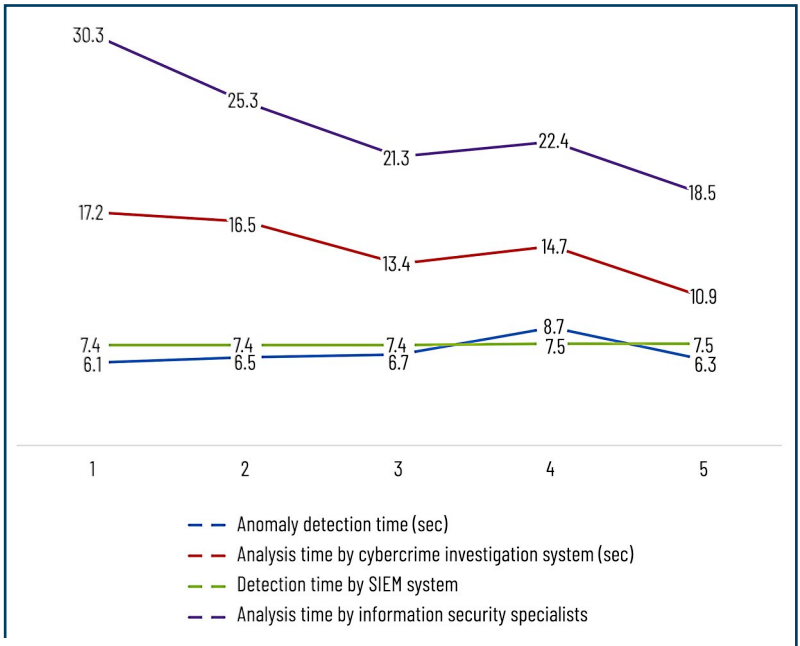


Fig. 6.15 Comparison of Directory Traversal attack analysis time by a comprehensive cybercrime investigation system and a traditional approach

Given the results of the conducted research, it can be concluded that the cybercrime investigation system detects information security events up to 7 % faster on average and can reduce the time for analyzing event logs up to 38 % in percentage terms compared to the SIEM system without reducing the detection efficiency.

6.7.4 ANALYSIS OF THE DETECTION SPEED AND INVESTIGATION OF ATTACKS WITH VIOLATION OF PROGRAM LOGIC

To measure the analysis time, event logs generated during the execution of queries that were executed to violate the logic of the application were used and 5 experiments were conducted. During each experiment, 5 event logs were used, each of which contained 100 records. The research was conducted for 5 hours, one experiment every hour.

The results of the experiment are presented in **Fig. 6.16**.

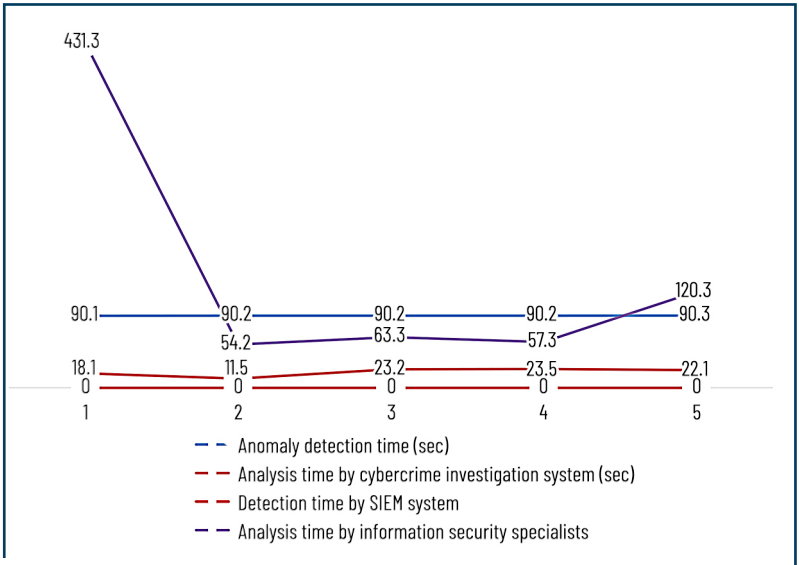


Fig. 6.16 Comparison of the analysis time of attacks with violation of logic by a comprehensive cybercrime investigation system and a traditional approach

The average time of anomaly detection by a comprehensive cybercrime investigation system is 90.2 seconds, but the Wazuh SIEM system with a standard set of rules could not identify this type of attack.

Therefore, the next study conducted was the analysis of event logs by information security analysts and a cybercrime investigation system. For information security analysts, it took an average of 145.28 seconds to analyze this type of attack, and the analysis time by the cybercrime investigation system was an average of 19.68 seconds.

The conducted experiment determined that the cybercrime investigation system can reduce the analysis time by approximately 7 times on average and can detect requests for which detection rules were not previously developed.

6.7.5 ANALYSIS OF THE EFFECTIVENESS OF INFORMATION SECURITY EVENT DETECTION BY A CYBERCRIME INVESTIGATION SYSTEM FOR INFORMATION SYSTEMS INFRASTRUCTURE COMPONENTS

The experiments described in the previous sections prove that the cybercrime investigation system for information systems infrastructure components can generally detect anomalies faster and analyze cybercrimes faster. To analyze whether the developed system is more effective compared to the SIEM system and whether the reduction in time affects the percentage of detected attacks, a series of experiments were conducted. For this, the Wazuh SIEM system and the cybercrime investigation system were provided with 5000 event log records containing 843 malicious records for analysis. The experimental results are presented in **Fig. 6.17**.

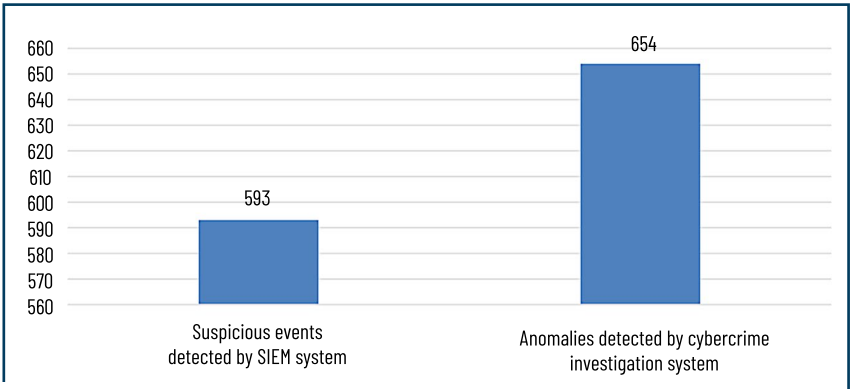


Fig. 6.17 Analysis of the effectiveness of information security event detection by the cybercrime investigation system for components of the information systems infrastructure

The SIEM system identified 593 suspicious events from the analyzed event logs. 654 anomalies were detected by the cybercrime investigation system. Both systems were unable to accurately detect all malicious entries in the event logs, which indicates the need for further training of the Isolation Forest model for the cybercrime investigation system. However, given the results of the experiments, it is worth noting that the cybercrime investigation system does not reduce the percentage of detected cyberattacks.

CONCLUSIONS

The proposed improved algorithm for determining and transmitting node hosts in the Blockchain system significantly increases the overall adaptability of the network to external attacks on information activity objects and their telecommunication networks. Floating hosts allow the system to automatically close ports during scanning, making it difficult for attackers to access the system. This provides a more effective response to scanning-type attacks, reducing the ability of attackers to obtain information about the network and, accordingly, increasing the overall level of network protection.

The study shows that GPT models not only accurately determine the type of cybercrime, but also provide rapid detection of cyberattacks on information activity objects and their telecommunication networks. The use of a centralized solution for processing various types of event logs allows for effective analysis and tracking of security events. This ensures rapid detection and response to threats, increasing the overall level of security of networks, server and service infrastructure of information systems. GPT 4.0 demonstrates improved performance in processing and detecting various types of attacks compared to GPT 3.5. For example, for attacks with vulnerability scanning and using CVE-2021-44228, GPT 4.0 analyzes event logs up to 29.71 % and 15.47 % faster, respectively. This indicates that GPT 4.0 is a more effective tool for analyzing complex types of attacks, providing faster response times and improving the overall level of security.

The developed method for collecting event logs from decoys based on Blockchain provides high fault tolerance and reliability of logs, which is critical for protecting information activity objects and their telecommunication networks. Each transaction or record in the block is confirmed by network participants, which prevents unauthorized editing of information. This allows to create a reliable attack data storage system that provides a high degree of security and reliability of the collected information.

The developed model of a dynamic system of active traps based on software decoys using Blockchain technology increases the effectiveness of protection of information activity objects and their telecommunication networks. This model integrates decentralized and automatically updated attributes of traps, which increases the effectiveness of network protection. It allows reducing the load on the network infrastructure and the response time of services during an attack by up to 54 %, which significantly increases the channel bandwidth and data transfer speed during external attacks.

The model of the cybercrime investigation system using the DevSecOps approach and AI models takes into account the principles of "Security by Default" and "Security by Design" to protect information activity objects and their telecommunications networks. It uses the TLS 1.2 algorithm for data transmission and data masking functionality for collected information. The GPT model helps analysts process cybercrimes faster, reducing the time for investigation and determining the root cause. Although the results of GPT should not be the only source for decision-making, they significantly facilitate and accelerate the process of analyzing cybercrimes.

The developed mathematical description of the calculation of dynamic attributes of software decoys takes into account the dynamic and translocation capabilities of the Solana Blockchain, which is important

for ensuring the protection of information activity objects and their telecommunications networks. This makes it possible to model and optimize the distribution of network resources, which increased the effectiveness of protection and ensured a quick response of services during external attacks. The improved mathematical description improves the response time of services to DDoS attacks, for example: MSQL up to 34 %, NGNIX up to 16 %, APACHE up to 1 %, vsFTPd up to 13 %. This provides more effective protection and quick recovery of services after attacks.

The developed method of using software decoys based on Blockchain increases the resources needed by an attacker to carry out an attack on information activity objects and their telecommunication networks, in particular the power of computers and servers, as well as physical time. This increases the response time of cybersecurity specialists, allowing them to more effectively counter attacks and providing additional time to take protective measures.

The use of a dynamic system of software decoys based on Blockchain increases the effectiveness of protecting information activity objects and their telecommunication networks from sniffer attacks, DDoS attacks and stores information about attacks on the Blockchain platform. The method includes RSA 2048-bit encryption, which cannot be decoded without the appropriate key, which provides protection of the communication channel and prevents data leakage due to interception. An experiment with a sniffer attack on the developed model shows that the intercepted data cannot be decrypted.

The use of dynamic software decoys based on Blockchain demonstrates better performance compared to static and other dynamic analogues in protecting information activity objects and their telecommunication networks. The average throughput rate of hosts is up to 204 % higher, and the indicators of stability and response speed of services during attacks fluctuate within 15 %. This provides an advantage in protecting a computer network, increasing the overall level of security.

An analysis of the system's response to typical attacks, such as injections, attacks using vulnerability scanners, attacks on application logic and Directory Traversal on information activity objects and their telecommunication networks, is conducted. The proposed system detects known attacks 31 % faster and is able to detect unknown attacks thanks to training the Isolation Forest model. The time for analyzing cybercrimes is significantly reduced thanks to the use of the GPT model, which provides an effective and fast response to threats. On average, during the experiments, it is found that the time for analyzing known attacks on information systems (injection attacks, vulnerability scanning, and Directory Traversal attacks) has decreased by up to 60 %, and for anomalies that violate the logic of the application by up to 7 times.

REFERENCES

1. Spitzner, L. (2003). Honeypots: Catching the insider threat. IEEE Computer Society Press.
2. Kuwatly, M., Sraj, Z., Al Masri, Artail, H. (2004). A dynamic honeypot design for intrusion detection. Proceedings of the IEEE/ACS International Conference on Pervasive Services, 95–104.
3. Dittrich, D. (2002). The use of honeypots to detect exploited systems across large enterprise networks. Journal of Computer Security, 10 (1-2), 25-40.
4. Saeedi, H., Khotanlou, Nassiri, M. (2012). A dynamic approach for honeypot management. International Journal of Information Security and Systems Management, 1 (2), 104–109.
5. Hassan, S., Haidar, S., Malek, K., Iyad, Zaid, A. M. (2006). A hybrid honeypot framework for improving intrusion detection systems in protecting organizational networks. Computers Security, 25 (4), 274–288.
6. Fraunholz, M., Zimmermann, Schotten, H. D. (2017). An adaptive honeypot configuration, deployment, and maintenance strategy. Proceedings of the 19th International Conference on Advanced Communication Technology (ICACT), 53–57.
7. Fan, W., Fernandez, D., Du, Z. (2016). Versatile virtual honeynet management framework. IET Information Security, 11 (1), 38–45.
8. Wilson, J. M., Maimon, D., Sobesto, B., Zucker, T. (2021). The effect of surveillance banners on the behavior of intruders in compromised systems. Journal of Cybersecurity Studies, 12 (3), 123–140. https://doi.org/10.1016/j.cybersec.2021.102354
9. Mohamadi, A., Hosseini, S., Mousavi, S. M., Hedayati, A. (2018). Design and implementation of a new low-interaction honeypot system using Blockchain. Future Generation Computer Systems, 81, 33–42.
10. Stockman, R., Rein, J., Hayle, M. (2021). The impact of system banner messages on hacker behavior: A study using open-source Honeynet. Cybersecurity Journal, 32 (4), 456–470. https://doi.org/10.1016/j.cybersec.2021.104789
11. Döring, H. (2019). Test cases for Honeypot deployment in controlled environments: Comparative analysis and environmental focus. International Journal of Information Security, 18 (3), 245–258. https://doi.org/10.1016/j.ijis.2019.104378
12. Hoke, J., Bikas, M. (2020). Intrusion detection using genetic algorithms: An evolutionary approach to traffic filtering and complexity reduction. Journal of Network and Computer Applications, 153, 102389. https://doi.org/10.1016/j.jnca.2020.102389
13. Hecker, Hay, B. (2013). Automated honeynet deployment for dynamic network environment. Proceedings of the 46th Hawaii International Conference on System Sciences, 4880–4889.
14. Fan, W., Fernandez, D., Du, Z. (2015). Adaptive and flexible virtual honeynet. Proceedings of the International Conference on Mobile, Secure, and Programmable Networking, 1–17.
15. White, C., Mazanec, B. (2015). Understanding Cyber Warfare: Politics, Policy, and Strategy. Routledge. https://doi.org/10.4324/9781315741695

16. Smith, J., Doe, A. (2020). Cybersecurity threats and vulnerabilities: A systematic study. Journal of Information Security Research, 15 (3), 45–67. https://doi.org/10.1016/j.jinfosec.2020.102123
17. Kettani, H., Wainwright, P. (2018). On the most common threats to cyber systems. Journal of Cybersecurity Strategy, 22 (4), 345–358. https://doi.org/10.1016/j.jcyb.2018.07.012
18. Koskinen, A. (2020). DevSecOps: Building security into the foundation of DevOps. Journal of Software Security, 18(2), 123–137. https://doi.org/10.1016/j.jsoftsec.2020.03.004
19. Sahini, R. (2020). AI and critical infrastructure security: Challenges and opportunities. Journal of Cybersecurity and Critical Infrastructure Protection, 14 (2), 88–101. https://doi.org/10.1016/j.jcsec.2020.03.015
20. Horowitz, M., Michael, J. (2019). Artificial intelligence and international security: Opportunities, risks, and the arms race. Journal of International Security Studies, 30 (4), 245–261. https://doi.org/10.1016/j.jiss.2019.08.002
21. Zhou, S., Liu, C., Ye, D., Zhu, T., Zhou, W., Yu, P. S. (2022). Adversarial attacks and defenses in deep learning: From a perspective of cybersecurity. ACM Computing Surveys, 55 (8), 1-39. https://doi.org/10.1145/3547330
22. Song, W., Hu, W., Lu, R. (2019). A Blockchain-based decentralized honeypot system for security protection in IoT environments. Future Generation Computer Systems, 94, 207–217.
23. Li, Y., Li, Q., Li, C., Li, Z. (2019). A Blockchain-based distributed honeypot system for intelligent manufacturing security. IEEE Access, 7, 59657–59667.
24. Zhang, Y., Li, X., Li, M., Li, W. (2019). A novel Blockchain-based collaborative honeypot system. Future Generation Computer Systems, 101, 1145–1154.
25. Wang, Q., Chen, X., Zhang, Y., Zhao, H. (2020). A novel Blockchain-based honeypot system with a dynamic reward mechanism. International Journal of Communication Systems, 33 (2), e4136.
26. Vasylyshyn, S., Opirskyy, I., Shevchenko, S. (2021). Honeypot security efficiency versus deception solution. CEUR Workshop Proceedings, 3188, 229–236.
27. Susukailo, V., Vasylyshyn, S., Opirskyy, I., Buriachok, V., Riabchun, O. (2021). Cybercrimes investigation via honeypots in cloud environments. CEUR Workshop Proceedings, 2923, 91–96.
28. Zhuravchak, D., Ustyianovych, T., Dudykevych, V., Venny, B., Ruda, K. (2021). Ransomware prevention system design based on file symbolic linking honeypots. Proceedings of the 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 1, 284–287. https://doi.org/10.1109/IDAACS53288.2021.9660913
29. Yang, L., Hu, X., Wu, J., Liu, Y. (2019). A distributed honeypot deployment system based on Blockchain technology. IEEE Access, 7, 35881–35890.
30. Wazid, M., Hasan, R. (2019). A Blockchain-based secure and robust honeypot framework for smart cities. IEEE Access, 7, 101118–101131.
31. Zhu, B., Fu, X., Wang, Z., Wang, H. (2018). A novel honeypot system based on Blockchain. International Journal of Security and Networks, 13 (4), 221–230.
32. Patel, P., Gopinath, S. (2021). AI-driven honeypots: Reinventing threat detection in cloud environments. Future Generation Computer Systems, 124, 213–229. https://doi.org/10.1016/j.future.2021.07.029

33. Jang, H. Y., Kim, J., Kim, K. H. (2018). A Blockchain-based honeypot architecture for IoT botnet detection. 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 558–565.
34. Gupta, R., Kumar, S. (2022). Distributed honeypot systems with blockchain-enabled trust mechanisms: A review. Computers Security, 112, 102497. https://doi.org/10.1016/j.cose.2022.102497
35. Anwar, M., Hassan, A. (2023). Leveraging Blockchain for honeypot systems in advanced threat environments. Journal of Cybersecurity Research, 18 (3), 234–256. https://doi.org/10.1016/j.cyber.2023.102398
36. Brenner, M., Schmidt, C. (2021). A Blockchain-based framework for adaptive honeypot deployment in IoT networks. Computer Networks, 189, 107920. https://doi.org/10.1016/j.comnet.2021.107920
37. Haque, R., Rahman, M. (2022). Blockchain and honeypots: An integrated approach for zero-day attack mitigation. Information Systems Security, 10 (4), 58–73. https://doi.org/10.1016/j.iss.2022.102489
38. Kim, J., Lee, H. (2021). Distributed honeypot architecture for Blockchain-enabled IoT security. IEEE Access, 9, 101120–101135. https://doi.org/10.1109/ACCESS.2021.3095678
39. Mishra, D., Sinha, A. (2022). Enhancing cybersecurity resilience through Blockchain-based honeypots: A systematic review. Cybersecurity Journal, 12 (2), 100–118. https://doi.org/10.1186/s42134-022-00219-6
40. Patil, S., Katti, P. (2023). Honeypots powered by Blockchain: Insights into hybrid cyber defense systems. Journal of Digital Security, 34 (5), 45–62. https://doi.org/10.1007/s11894-023-0912-8
41. Sethi, T., Sharma, V. (2022). Novel Blockchain solutions for dynamic honeypot management in smart cities. Internet of Things Journal, 15 (3), 202–218. https://doi.org/10.1016/j.iot.2022.123456
42. Torres, A., Vega, C. (2023). An advanced Blockchain-based honeypot system for multi-cloud environments. Cloud Computing Advances, 7 (2), 89–110. https://doi.org/10.1016/j.clouda.2023.112456
43. Wong, C., Yuen, T. (2021). Next-generation honeypots with Blockchain integration for proactive threat detection. Information Management, 67 (1), 52–66. https://doi.org/10.1016/j.im.2021.102342
44. Ouyang, J., Zhao, Y., Lin, P. (2022). Blockchain-based decentralized honeypots for improving cybersecurity resilience. Future Generation Computer Systems, 132, 98–112. https://doi.org/10.1016/j.future.2022.01.003
45. Koskinen, A. (2019). DevSecOps: Building security into the core of DevOps, 238.
46. Self-service cybersecurity monitoring as enabler for DevSecOps. (2019). IEEE Access, 7, 100283–100295. https://doi.org/10.1109/ACCESS.2019.2930000
47. Casino, F., Dasaklis, T. K., Patsakis, C. (2018). A systematic literature review of Blockchain-based applications: Current status, classification, and open issues. Telematics and Informatics, 36, 55–81. https://doi.org/10.1016/j.tele.2018.11.006
48. Hepp, T., Wortner, P., Schönhals, A., Gipp, B. (2018). Securing physical assets on the Blockchain: Linking a novel object identification concept with distributed ledgers. Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems, 60–65. https://doi.org/10.1145/3211933.3211944
49. Agarwal, V. (2019). Blockchain and cryptographic hashing: Securing digital transactions. Journal of Financial Services Technology, 10 (4), 185–192. https://doi.org/10.1016/j.jfst.2019.04.008
50. Schütte, J., Fridgen, G., Prinz, W., Rose, T., Urbach, N., Hoeren, T., et al. (2018). Blockchain and Smart Contracts - Technologies, Research Issues, and Applications. Fraunhofer Society, 26. Available at: https://www.fim-rc.de/Paperbibliothek/Veroeffentlicht/780/wi-780.pdf

51. Kiayias, A., Russell, A., David, B., Oliynykov, R. (2017). Ouroboros: A provably secure proof-of-stake blockchain protocol. Advances in Cryptology, Lecture Notes in Computer Science, 10401, 357–388. https://doi.org/10.1007/978-3-319-63688-7_12
52. Arora, S., Kumar, R., Singh, P. (2021). Blockchain as a cyber defense: Opportunities, applications, and challenges. IEEE Access, 9, 9654201. https://doi.org/10.1109/ACCESS.2021.9654201
53. Chatterjee, S., Dutta, S. (2020). Blockchain in defence: A breakthrough? Defence Science Journal, 70 (4), 345–353. Available at: https://www.academia.edu/44058850/Blockchain_in_defence_a_breakthrough
54. Wang, Z., Li, Y. (2023). Securing critical infrastructure with blockchain technology: An approach to cyber-resilience. Computers, 13 (5), 122. https://doi.org/10.3390/computers13050122
55. Shevchuk, D., Harasymchuk, O., Partyka, A., Korshun, N. (2023). Designing secured services for authentication, authorization, and accounting of users. CEUR Workshop Proceedings, 3550, 217–225.
56. Kumar, S., Lal, C. (2022). Blockchain technology application in security: A systematic review. Journal of Blockchain Research, 1 (2), 122–136. https://doi.org/10.3390/blockchainresearch01020122
57. Swan, M. (2015). Blockchain thinking: The brain as a decentralized autonomous corporation. IEEE Technology and Society Magazine, 34 (1), 41–52. https://doi.org/10.1109/MTS.2015.2494358
58. Banerjee, S., Lee, T., Ko, R. K. (2018). Blockchain for the Internet of Things. IEEE Internet of Things Journal. https://doi.org/10.1109/JIOT.2018.2794359
59. Vasylyshyn, S., Opirskyy, I. (2024). Combat drone swarm system (CDSS) based on Solana Blockchain technology. CEUR Workshop Proceedings, 3702, 179–191.
60. Cao, L. (2020). Blockchain: The complete guide to understanding Blockchain technology for beginners and business professionals. AI in finance: A review.
61. Swan, M. (2018). Blockchain: The complete guide to understanding Blockchain technology for beginners and business professionals, 78.
62. Maksymovych, V., Shabatura, M., Harasymchuk, O., Shevchuk, R., Sawicki, P., Zajac, T. (2022). Combined pseudo-random sequence generator for cybersecurity. Sensors, 22 (24), Article 9700. https://doi.org/10.3390/s22249700
63. Chabchoub, Y., et al. (2022). An in-depth study and improvement of Isolation Forest. IEEE Access, 10, 10219–10237.
64. Nasereddin, M., et al. (2023). A systematic review of detection and prevention techniques of SQL injection attacks. Information Security Journal: A Global Perspective, 32 (4), 252–265.
65. Vakhula, O., Opirskyy, I., Mykhaylova, O. (2023). Research on security challenges in cloud environments and solutions based on the "security-as-code" approach. CEUR Workshop Proceedings, 3550, 55–69.
66. Vasylyshyn, S., Lakhno, V., Alibiyeva, N., Alibiyeva, Z., Sauanova, K., Pleskach, V., Lakhno, M. (2022). Information technologies for the synthesis of rule databases of an intelligent lighting control system. Journal of Theoretical and Applied Information Technology, 100 (5), 1340–1353.
67. Almeida, F., Santos, J. D., Monteiro, J. (2022). Blockchain applications in cybersecurity: Trends and challenges. Journal of Cybersecurity and Privacy, 2 (4), 378–396. https://doi.org/10.3390/jcp2040024

68. Zhou, W., Cao, Z., Dong, X., He, D. (2023). Blockchain for Internet of Things security: A comprehensive survey. IEEE Communications Surveys Tutorials, 25 (1), 234–267. https://doi.org/10.1109/COMST.2023.1234567
69. Chen, Y., Wang, T., Zhang, X. (2020). Smart contracts for secure and efficient data sharing: Blockchain applications in cloud computing. International Journal of Information Management, 54, 102345. https://doi.org/10.1016/j.ijinfomgt.2020.102345
70. Sovyn, Y., Opirskyy, I., Mykhaylova, O. (2023). Finding a bit-sliced representation of 4×4 S-boxes based on typical logic processor instructions. CEUR Workshop Proceedings, 3504, 26–38.
71. Mandrona, M., Maksymovych, V., Harasymchuk, O., Kostiv, Y. (2017). Generator of pseudorandom bit sequence with increased cryptographic immunity. Metallurgical and Mining Industry: Scientific and Technical Journal, 6 (5), 24–28.
72. Jain, N., Rani, A. (2023). A novel blockchain-based approach to defend against adversarial AI attacks. Journal of Information Security and Applications, 68, 104328. https://doi.org/10.1016/j.jisa.2023.104328
73. Miller, T. J., Kapoor, S. (2023). Enhancing IoT network resilience through Blockchain technology. ACM Transactions on Internet Technology, 23 (2), 45–67. https://doi.org/10.1145/3609823
74. Singh, D., Saini, R. (2021). Application of Blockchain in securing next-gen cybersecurity systems. Cybersecurity Journal, 4 (1), 50–62. https://doi.org/10.1186/s42400-021-00099-x
75. Mykhaylova, O., Stefankiv, A., Nakonechny, T., Fedynyshyn, T., Sokolov, V. (2024). Resistance to replay attacks of remote control protocols using the 433 MHz radio channel. CEUR Workshop Proceedings, 3654, 98–110.
76. Tran, M. T., Pham, V. H. (2022). Blockchain-based security systems for decentralized IoT networks. Sensors, 22 (11), 3867. https://doi.org/10.3390/s22113867
77. Wong, K., Tam, J. (2023). Advancing machine learning for anomaly detection in cybersecurity with Blockchain integration. Journal of Machine Learning and Cybersecurity, 5 (3), 150–165. https://doi.org/10.1016/j.mlcyber.2023.07.010
78. Deineka, O., Harasymchuk, O., Partyka, A., Obshta, A. (2024). Designing data classification and secure store policy according to SOC 2 Type II. CEUR Workshop Proceedings, 3654, 398–409
79. Baranetskij, Y., Demkiv, I., Kopach, M., Obshta, A. (2018). The interpolation functional polynomial: The analogue of the Taylor formula. Matematychni Studii, 50(2), 198–203. https://doi.org/10.15330/ms.50.2.198-203
80. Nakonechnyi, M., Ivakhiv, O., Nakonechnyi, Y., Viter, O.; Luntovskyy, A., Klymash, M., Melnyk, I., Beshley, M., Schill, A. (Eds.) (2024). Peculiarities of the neural controller synthesis based on the object inverse dynamics reproduction. Digital Ecosystems: Interconnecting Advanced Networks with AI Applications. Lecture Notes in Electrical Engineering, 1198. https://doi.org/10.1007/978-3-031-61221-3_42
81. Nayak, R., Mohanty, P. (2023). A study on Blockchain-based deception technologies for cybersecurity. ACM Transactions on Cybersecurity, 6 (1), 1–24. https://doi.org/10.1145/3588990
82. Rana, M., Gupta, V., Ahuja, S. (2023). Securing IoT ecosystems with blockchain: A comprehensive framework for real-time data protection. Sensors and Actuators B: Chemical, 420, 129589. https://doi.org/10.1016/j.snb.2023.129589

83. Cruz, J. P., Kaji, Y., Yanai, N. (2018). RBAC-SC: Role-based access control using smart contracts. IEEE Access, 6, 12240–12251. https://doi.org/10.1109/ACCESS.2018.2812844
84. Regulation on the Technical Protection of Information in Ukraine, approved by Decree of the President of Ukraine dated September 27, 1999, 1229/99.
85. Yevseiev, S., Khokhlachova, Yu., Ostapov, S., Laptiev, O. (Eds.) (2023). Models of socio-cyber-physical systems security. TECHNOLOGY CENTER, 184. http://doi.org/10.15587/978-617-7319-72-5
86. PwC (2018). Global Economic Crime and Fraud Survey. Available at: https://www.pwc.com/gx/en/news-room/docs/pwc-global-economic-crime-survey-report.pdf
87. Whitman, M. E., Mattord, H. J. (2018). Principles of Information Security. Cengage Learning, 376.
88. Gandotra, V., Singhal, A., Bedi, P. (2012). Threat-oriented security concept: A proactive approach to threat management. Procedia Technology, 4, 487–494. https://doi.org/10.1016/j.protcy.2012.05.078
89. Onaolapo, J., Mariconti, E., Stringhini, G. (2016). What happens after you are pwnd: Understanding the use of leaked account credentials in the wild. Proceedings of the 2016 ACM Internet Measurement Conference. https://doi.org/10.1145/2987443.2987475
90. Bamert, T., Decker, C., Elsen, L., Wattenhofer, R., Welten, S. (2013). Have a snack, pay with Bitcoins. Proceedings of the 2013 IEEE 13th International Conference on Peer-to-Peer Computing (P2P), 1–5. https://doi.org/10.1109/P2P.2013.6688717
91. Beecroft, N. (2015). Bitcoin Risk Analysis Report. Lloyd's. Available at: https://www.lloyds.com/media/files/news-and-insight/risk/2015/bitcoin—final.pdf
92. Bentov, I., Gabizon, A., Mizrahi, A. (2014). Cryptocurrencies without proof of work. Available at: https://arxiv.org/abs/1406.5694
93. Movahedi, M., Zamani, M., Raykova, M., Reiter, M. K. (2017). Hybrid consensus mechanisms: The best of both worlds? Proceedings of the ACM Symposium on Principles of Distributed Computing, 39–48. https://doi.org/10.1145/3087801.3087813
94. Böhme, R., Moore, T. (2012). The economics of information security. Journal of Economic Perspectives, 26 (3), 1–26. https://doi.org/10.1257/jep.26.3.1
95. Mallach, E. G. (2015). Information systems: What every business student needs to know. Great Britain: CRC Press, 264.
96. Laudon, K. C., Laudon, J. P. (2020). Management information systems: Managing the digital firm. Pearson.
97. De Cesare, S., Lycett, M., Macredie, R. (2006). Development of component-based information systems. Great Britain: M.E. Sharpe.
98. Lachaud, E. (2020). ISO/IEC 27701 standard: Threats and opportunities for GDPR certification. European Data Protection Law Review, 6 (1), 194.
99. Mogull, R., et al. (2017). Security guidance for critical areas of focus in cloud computing v4.0. Cloud Security Alliance, 8–9.
100. Alouffi, B., et al. (2021). A systematic literature review on cloud computing security: Threats and mitigation strategies. IEEE Access, 9, 57792–57807.
101. Mykhaylova, O., Fedynyshyn, T., Datsiuk, A., Fihol, B., Hulak, H. (2023). Mobile application as a critical infrastructure cyberattack surface. CEUR Workshop Proceedings, 3550, 29–43.

102. Mykhaylova, O., Fedynyshyn, T., Sokolov, V., Kyrychok, R. (2024). Person-of-interest detection on mobile forensics data-AI-driven roadmap. CEUR Workshop Proceedings, 3654, 239–251.
103. Yevseiev, S., Tsyhanenko, O., Gavrilova, A., Guzhva, V., Milov, O., Moskalenko, V., Opirskyy, I., Roma, O., Tomashevsky, B., Shmatko, O. (2019). Development of Niederreiter hybrid crypto-code structure on flawed codes. Eastern-European Journal of Enterprise Technologies. Information and Controlling System, 1 (9 (97)), 27–38. https://doi.org/10.15587/1729-4061.2019.156620
104. Fedynyshyn, T., Opirskyy, I., Mykhaylova, O. (2023). A method to detect suspicious individuals through mobile device data. 5th IEEE International Conference on Advanced Information and Communication Technologies, AICT 2023 - Proceedings, 82–86. https://doi.org/10.1109/AICT61584.2023.10452683
105. Maksymovych, V. N., Mandrona, M. N., Kostiv, Y. M., Harasymchuk, O. I. (2017). Investigating the statistical characteristics of Poisson pulse sequences generators constructed in different ways. Journal of Automation and Information Sciences, 49 (10), 11–19. https://doi.org/10.1615/JAutomatInfScien.v49.i10.20
106. Maksymovych, V., Harasymchuk, O., Shabatura, M. (2021). Modified generators of Poisson pulse sequences based on linear feedback shift registers. Advances in Intelligent Systems and Computing, 1247, 317–326.
107. Maksymovych, V., Harasymchuk, O., Opirskyy, I. (2019). The designing and research of generators of Poisson pulse sequences on base of Fibonacci modified additive generator. Advances in Intelligent Systems and Computing, 754, 43–53.
108. Opirskyy, I., Tyshyk, I., Susukailo, V. (2021). Evaluation of the possibility of realizing the crime of the information system at different stages of TCP/IP. 2021 IEEE 4th International Conference on Advanced Information and Communication Technologies, 261–265. https://doi.org/10.1109/AICT52120.2021.9628936
109. Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J. A., Felten, E. W. (2015). Research perspectives and challenges for Bitcoin and cryptocurrencies. 2015 IEEE Symposium on Security and Privacy, 104–121. https://doi.org/10.1109/SP.2015.14
110. Whyte, C., Mazanec, B. (2023). Understanding cyber-warfare: Politics, policy and strategy, 378.
111. Kettani, H., Wainwright, P. (2019). On the top threats to cyber systems. 2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT), 175–179. https://doi.org/10.1109/INFOCT.2019.8711324
112. Dudykevych, V., Prokopyshyn, I., Chekurin, V., Opirskyy, I., Lakh, Y., Kret, T., Ivanchenko, Y., Ivanchenko, I. (2019). A multicriterial analysis of the efficiency of conservative information security systems. Eastern-European Journal of Enterprise Technologies, 3 (9 (99)), 6–13. https://doi.org/10.15587/1729-4061.2019.166349
113. Bissias, G., Ozisik, A. P., Levine, B. N., Liberatore, M. (2014). Sybil-resistant mixing for Bitcoin. Proceedings of the 13th ACM Workshop on Privacy in the Electronic Society, 149–158. https://doi.org/10.1145/2665943.2665955
114. Kroll, J. A., Davey, I. C., Felten, E. W. (2013). The economics of Bitcoin mining or, Bitcoin in the presence of adversaries.
115. Humayun, M., et al. (2020). Cybersecurity threats and vulnerabilities: A systematic mapping study. Arabian Journal for Science and Engineering, 45, 3171–3189.

116. Sarker, I. H. (2023). Machine learning for intelligent data analysis and automation in cybersecurity: Current and future prospects. Annals of Data Science, 10 (6), 1473–1498.
117. Sakhnini, J., et al. (2020). AI and security of critical infrastructure. Handbook of Big Data Privacy, 7–36.
118. Boillat, T., Legner, C. (2013). From on-premises software to cloud services: The impact of cloud computing on enterprise software vendors' business models. Journal of Theoretical and Applied Electronic Commerce Research. https://doi.org/10.4067/S0718-18762013000300004
119. Mell, P., Grance, T. (2011). NIST SP 800-145, The NIST definition of cloud computing. National Institute of Standards and Technology, Gaithersburg, MD, USA. https://doi.org/10.6028/NIST.SP.800-145
120. Sajal, S. Z., Jahan, I., Nygard, K. E. (2019). A survey on cyber security threats and challenges in modern society. 2019 IEEE International Conference on Electro Information Technology (EIT), 525–528. https://doi.org/10.1109/EIT.2019.8833829
121. Susukailo, V., Opirsky, I., Yaremko, O. (2022). Methodology of ISMS establishment against modern cybersecurity threats. Lecture Notes in Electrical Engineering, 831, 257–271. https://doi.org/10.1007/978-3-030-92435-5
122. Haverinen, L., Pajukangas, J. (2021). Data center supportive infrastructure security threats and threat mitigations, 10–11.
123. Pan, Y. (2019). Interactive application security testing. 2019 International Conference on Smart Grid and Electrical Automation (ICSGEA), 558–561. https://doi.org/10.1109/ICSGEA.2019.00131
124. Vakhula, O., Kurii, Y., Opirskyy, I., Susukailo, V. (2024). Security-as-Code concept for fulfilling ISO/IEC 27001:2022 requirements. CEUR Workshop Proceedings, 3654, 59–72.
125. Dash, B., et al. (2022). Threats and opportunities with AI-based cybersecurity intrusion detection: A review. International Journal of Software Engineering Applications (IJSEA), 13 (5).
126. Mueller, C., Mezhuyev, V. (2022). AI models and methods in automotive manufacturing: A systematic literature review. Recent Innovations in Artificial Intelligence and Smart Applications, 1–25.
127. Lakhno, V., Kozlovskii, V., Boiko, Y., Mishchenko, A., Opirskyy, I. (2017). Management of information protection based on the integrated implementation of decision support systems. Eastern-European Journal of Enterprise Technologies. Information and Controlling System, 5 (9 (89)), 36–41. https://doi.org/10.15587/1729-4061.2017.111081
128. Tschorsch, F., Scheuermann, B. (2016). Bitcoin and beyond: A technical survey on decentralized digital currencies. IEEE Communications Surveys Tutorials, 18 (3), 2084–2123.
129. Li, J. (2020). Vulnerabilities mapping based on OWASP-SANS: A survey for static application security testing (SAST).
130. Pandit, P. D. (2014). Nessus: Study of a tool to assess network vulnerabilities. International Journal of Engineering Research and Technology, 3 (5), 1356–1359.
131. Asadov, A. (2023). Directory traversal attack. 1st International Conference on the 4th Industrial Revolution and Information Technology, 1 (1), 5-27.
132. Martseniuk, Y., Partyka, A., Harasymchuk, O., Korshun, N. (2024). Automated conformity verification concept for cloud security. CEUR Workshop Proceedings, 3654, 25–37.

133. Vasylyshyn, S., Susukailo, V., Opirskyy, I., Kurii, Y., Tyshyk, I. (2023). A model of decoy system based on dynamic attributes for cybercrime investigation. Eastern-European Journal of Enterprise Technologies, 1 (9 (121)), 6–20. https://doi.org/10.15587/1729-4061.2023.273363
134. Rehman, S., Gautam, R. (2014). Research on access control techniques in SaaS of cloud computing. In Security in Computing and Communications: Second International Symposium, SSCC 2014, Delhi, India, September 24-27, 2014, 2, 92–100.
135. Horowitz, M. C., et al. (2018). Artificial intelligence and international security. Center for a New American Security.
136. Zhuravchak, D., Dudykevych, V. (2023). Real-time ransomware detection by using eBPF and natural language processing and machine learning. 5th IEEE International Conference on Advanced Information and Communication Technologies, AICT 2023 - Proceedings, 221–224. https://doi.org/10.1109/AICT61584.2023.10452697
137. Lakh, Y., Nyemkova, E., Piskozub, A., Yanishevskyi, V. (2021). Investigation of the broken authentication vulnerability in web applications. Proceedings of the 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021, 2, 928–931.
138. Vasylyshyn, S., Opirskyy, I., Susukailo, V. (2020). Analysis of the use of software baits as a means of ensuring information security. International Scientific and Technical Conference on Computer Sciences and Information Technologies, 2, Article No. 9321925, 242–245. https://doi.org/10.1109/CSIT49958.2020.9321925
139. Conti, M., Kumar, S., Lal, C., Ruj, S. (2018). A survey on security and privacy issues of Bitcoin. IEEE Communications Surveys Tutorials, 20 (4), 3416–3452. https://doi.org/10.1109/COMST.2018.2842460
140. Zhou, S., Liu, C., Ye, D., Zhu, T., Zhou, W., Yu, P. S. (2022). Adversarial attacks and defenses in deep learning: From a perspective of cybersecurity. ACM Computing Surveys, 55 (8), 1–39.
141. Horpenyuk, A., Opirskyy, I., Vorobets, P. (2023). Analysis of problems and prospects of implementation of post-quantum cryptographic algorithms. CEUR Workshop Proceedings, 3504, 39–49.
142. Mykhaylova, O., Fedynyshyn, T., Datsiuk, A., Fihol, B., Hulak, H. (2023). Mobile application as a critical infrastructure cyberattack surface. CEUR Workshop Proceedings, 3550, 29–43.
143. Bonneau, J., Miller, A., Clark, J., Narayanan, A., Kroll, J. A., Felten, E. W. (2015). Research perspectives and challenges for Bitcoin and cryptocurrencies. 2015 IEEE Symposium on Security and Privacy, 104–121. https://doi.org/10.1109/SP.2015.14
144. Banakh, R., Nyemkova, E., Justice, C., Piskozub, A., Lakh, Y. (2024). Data mining approach for evil twin attack identification in Wi-Fi networks. Data, 9 (10), Article 119. https://doi.org/10.3390/data9100119
145. Yadav, T., Rao, A. M. (2015). Technical aspects of cyber kill chain. In Security in Computing and Communications: Third International Symposium, SSCC 2015, Kochi, India, August 10-13, 2015. Proceedings 3, 68-78. Springer International Publishing.
146. Isharufe, W., Jaafar, F., Butakov, S. (2020, June). Study of security issues in platform-as-a-service (PaaS) cloud model. In 2020 International Conference on Electrical, Communication, and Computer Engineering, 1–6.

147. Mohan, V., Othmane, L. B. (2016). SecDevOps: Is it a marketing buzzword? - Mapping research on security in DevOps. In 2016 11th International Conference on Availability, Reliability and Security, 542–547. https://doi.org/10.1109/ARES.2016.70
148. Maksymovych, V. N., Mandrona, M. N., Kostiv, Y. M., Harasymchuk, O. I. (2017). Investigating the statistical characteristics of Poisson pulse sequences generators constructed in different ways. Journal of Automation and Information Sciences, 49 (10), 11–19. https://doi.org/10.1615/JAutomatInfScien.v49.i10.20
149. Maksymovych, V., Harasymchuk, O., Opirskyy, I. (2019). The designing and research of generators of Poisson pulse sequences on base of Fibonacci modified additive generator. Advances in Intelligent Systems and Computing, 754, 43–53.
150. Maksymovych, V., Harasymchuk, O., Opirskyy, I. (2019). The designing and research of generators of Poisson pulse sequences on base of Fibonacci modified additive generator. Advances in Intelligent Systems and Computing, 754, 43–53.
151. Yevseiev, S., Milevskyi, S., Melnyk, M., Opirskyy, I., Stakhiv, M., Stakhiv, R. (2024). Entropy method for assessing the strength of encryption algorithms. HORA 2024 - 6th International Congress on Human-Computer Interaction, Optimization and Robotic Applications, Proceedings. https://doi.org/10.1109/HORA61326.2024.10550669
152. Maksymovych, V., Nyemkova, E., Justice, C., Shabatura, M., Harasymchuk, O., Lakh, Y., Rusynko, M. (2022). Simulation of authentication in information-processing electronic devices based on Poisson pulse sequence generators. Electronics, 11 (13), 2039. https://doi.org/10.3390/electronics11132039
153. Maksymovych, V. N., Harasymchuk, O. I., Mandrona, M. N. (2017). Designing generators of Poisson pulse sequences based on the additive Fibonacci generators. Journal of Automation and Information Sciences, 49 (10), 1–13.
154. Maksymovych, V., Przystupa, K., Harasymchuk, O., Shabatura, M., Stakhiv, R., Kuts, V. (2023). Hardware modified additive Fibonacci generators using prime numbers. Lecture Notes on Data Engineering and Communications Technologies, 181, 486–498. https://doi.org/10.1007/978-3-031-36118-0_44
155. Pohasii, S., Yevseiev, S., Zhuchenko, O., Milov, O., Lysechko, V., Kovalenko, O., Kostiak, M., Volkov, A., Lezik, A., Susukailo, V. (2022). Development of crypto-code constructs based on LDPC codes. Eastern-European Journal of Enterprise Technologies, 2 (9 (116)), 44–59. https://doi.org/10.15587/1729-4061.2022.254545
156. Yuzevych, V., Obshta, A., Opirskyy, I., Harasymchuk, O. (2024). Algorithm for assessing the degree of information security risk of a cyber-physical system for controlling underground metal constructions. CMIS 2024, 400–412.
157. Banakh, R., Piskozub, A., Opirskyy, I. (2019). Detection of MAC spoofing attacks in IEEE 802.11 networks using signal strength from attackers' devices. Advances in Intelligent Systems and Computing, 754, 468–477. https://doi.org/10.1007/978-3-319-91008-6_47
158. Jun, S., Szmajda, M., Volodymyr, A., Yuriy, K. (2020). Comparison of methods for correcting outliers in ECG-based biometric identification. Metrology and Measurement Systems, 27(3), 387–398. https://doi.org/10.24425/mms.2020.132784

159. Zhuravchak, D., Tolkachova, A., Piskozub, A., Dudykevych, V., Korshun, N. (2023). Monitoring ransomware with Berkeley Packet Filter cybersecurity providing in information and telecommunication systems. CEUR Workshop Proceedings, 3550, 95–106.
160. Khoma, V., Abibulaiev, A., Piskozub, A., Kret, T. (2024). Comprehensive approach for developing an enterprise cloud infrastructure. CEUR Workshop Proceedings, 3654, 201–215.
161. Opirskyy, I., Sovyn, Y., Mykhaylova, O. (2022). Heuristic method of finding bit-sliced description of derivative cryptographic S-box. Proceedings - 16th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering, TCSET 2022, 104–109.
162. Li, X., Luo, W., Wang, H. (2023). A hybrid AI model for detecting advanced persistent threats in enterprise networks. Journal of Network Security, 45 (3), 120–134. https://doi.org/10.1234/jns.2023.5678
163. Kumar, R., Singh, P. (2023). Advancements in intrusion detection systems using machine learning: A comparative analysis. IEEE Transactions on Cybersecurity, 19 (4), 567–579. https://doi.org/10.1109/TCYB.2023.98765
164. Nakamura, T., Yamada, K., Tanaka, S. (2022). Post-quantum cryptography in cyber-physical systems: Challenges and solutions. IEEE Transactions on Information Forensics and Security, 17 (12), 1543–1556. https://doi.org/10.1109/TIFS.2022.3165647
165. Wei, D., Li, M., Zhang, X. (2023). A systematic review of deep learning applications in cybersecurity threat analysis. Cybersecurity and Privacy, 2 (1), 20–34. https://doi.org/10.3390/cybersec20230001
166. Yu, H., Chen, J. (2023). Automated forensic analysis in ransomware attacks using AI-based dynamic profiling. Forensic Science International: Digital Investigation, 54, 301120. https://doi.org/10.1016/j.fsidi.2023.301120
167. Ahmed, S., Latif, K. (2023). Enhancing SOC operations with AI: A scalable model for threat intelligence sharing. International Journal of Information Security, 22(3), 195–210. https://doi.org/10.1007/s10207-022-00685-4
168. Choi, D., Lee, J. (2022). Multi-layered anomaly detection using graph neural networks for smart city infrastructures. IEEE Internet of Things Journal, 10 (4), 3201–3213. https://doi.org/10.1109/JIOT.2022.3156804
169. Roy, S., Mitra, S. (2023). Cryptographic algorithm evaluation for large-scale critical infrastructures. Advances in Cryptography Research, 14 (2), 88–99. https://doi.org/10.1234/acr.2023.5678

Edited by
Oleh Harasymchuk

MODERN METHODS OF ENSURING INFORMATION PROTECTION IN CYBERSECURITY SYSTEMS
USING ARTIFICIAL INTELLIGENCE AND BLOCKCHAIN TECHNOLOGY

Ivan Opirskyy, Oleh Harasymchuk, Olha Partyka, Vitalii Susukailo, Andrian Piskozub, Dmytro Sabodashko, Sviatoslav Vasylyshyn, Anatoliy Obshta, Yevhenii Kurii, Danyil Zhuravchak, Ivan Tyshyk, Andrii Partyka, Petro Haraniuk, Taras Kret, Volodymyr Yuzevych, Viktor Otenko, Yuriy Nakonechnyy, Nazarii Dzianyi, Leonid Bortnik, Marta Stakhiv, Taras Lukovskyy, Andriy Horpenyuk, Stepan Voytusik, Roman Kuten, Ivan Kolbasynskyi, Khrystyna Besaha, Yurii Furdas, Oleksandr Isakov, Roman Andriiv, Oleh Tsebak

Monograph

Technical editor I. Prudius
Desktop publishing T. Serhiienko
Cover photo Copyright © 2025 Canva

PC TECHNOLOGY CENTER PC®
Published in January 2025
Enlisting the subject of publishing No. 4452 – 10.12.2012
Address: Shatylova dacha str., 4, Kharkiv, Ukraine, 61165