Edited by
Oleh Harasymchuk

INTELLIGENT CYBER DEFENCE SYSTEMS: DETECTION OF RANSOMWARE AND PROTECTION OF WIRELESS NETWORKS BASED ON ARTIFICIAL INTELLIGENCE TECHNOLOGIES

Monograph

Technology Center

2025

UDC 004.056
M78

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

Approved by the Academic Council of Lviv Polytechnic National University, Protocol No. 5 of 01.03.2025

Reviewers:
Dudykevych Valerii, Doctor of Technical Science, Professor, Head of the Department of Information Security of Lviv Polytechnic National University;
Korchenko Alexandr, Doctor of Technical Sciences, Professor, Head of the Department of Information Technology Security of National Aviation University.

M78 **Authors:**
Edited by Oleh Harasymchuk
Ivan Opirskyy, Roman Banakh, Danyil Zhuravchak, Oleh Harasymchuk, Olha Partyka, Andrian Piskozub, Elena Nyemkova, Sviatoslav Vasylyshyn, Andrii Partyka, Yuriy Nakonechnyy, Taras Lukovskyy, Vitalii Susukailo, Viktor Otenko, Ivan Tyshyk, Nazarii Dzianyi, Dmytro Sabodashko, Petro Haraniuk, Valerii Dudykevych, Serhiy Semenyuk, Marta Stakhiv, Ihor Zhuravel, Taras Kret, Lesya Mychuda, Zynoviy Mychuda, Orest Polotai, Yevhenii Kurii, Nataliya Nakonechna, Nataliya Luzhetska, Anatoliy Obshta, Tetiana Korobeinikova
Intelligent cyber defence systems: detection of ransomware and protection of wireless networks based on artificial intelligence technologies: monograph / O. Harasymchuk and others. – Kharkiv: TECHNOLOGY CENTER PC, 2025. – 182 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 200 items.

DOI: 10.15587/978-617-8360-22-1
ISBN 978-617-8360-22-1 (on-line)

Cite as: Harasymchuk, O. (Ed.) (2025). Intelligent cyber defence systems: detection of ransomware and protection of wireless networks based on artificial intelligence technologies: collective monograph. Kharkiv: TECHNOLOGY CENTER PC, 182. doi: http://doi.org/10.15587/978-617-8360-22-1

9 786178 360221

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

ROMAN BANAKH
PhD, Senior Lecturer
Department of Information Security Technologies
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0001-6897-8206

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

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

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

ELENA NYEMKOVA
Doctor of Technical Sciences, Professor
Department of the Information Security Technologies
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0003-0690-2657

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

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

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

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

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

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

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

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

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

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

VALERII DUDYKEVYCH
Doctor of Technical Sciences, Professor
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0001-8827-9920

SERHIY SEMENYUK
PhD, Associate Professor
Department of Information Technology Security
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0002-8143-5887

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

IHOR ZHURAVEL
Doctor of Technical Sciences, Senior Research, Head of Department
Department of Information Technology Security
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0003-1114-0124

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

LESYA MYCHUDA
Doctor of Technical Sciences, Professor
Department of the Information Security Technologies
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0001-8266-1782

ZYNOVIY MYCHUDA
Doctor of Technical Sciences, Professor
Department of the Computerized Automatic Systems
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0002-3317-5195

OREST POLOTAI
PhD, Associate Professor
Department of Information Security Management
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0003-4593-8601

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

NATALIYA NAKONECHNA
PhD, Associate Professor
Department of Information Technology Security
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0003-1377-4315

NATALIYA LUZHETSKA
Senior Lecturer
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0001-5151-312X

ANATOLIY OBSHTA
Doctor of Technical Sciences, Professor
Department of Information Protection
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0009-0001-2450-5439

TETIANA KOROBEINIKOVA
PhD, Associate Professor
Department of Information Security Technologies
Lviv Polytechnic National University
ORCID ID: https://orcid.org/0000-0003-2487-8742

ABSTRACT

This monograph is devoted to a comprehensive study of two critical areas of cybersecurity: countering ransomware and protecting IEEE 802.11 wireless networks. The work combines theoretical research and practical solutions for creating effective information protection systems.

The first part of the monograph explores methods for detecting and countering ransomware viruses in real time using eBPF technology and machine learning models. It presents an innovative model of an integrated data collection system that combines monitoring of system calls, file and cryptographic activity with network traffic analysis. A comprehensive classification model based on an ensemble of decision trees and random forests is proposed, demonstrating malware detection accuracy above 95%. A methodology for applying deep neural networks to identify complex behaviour patterns of ransomware viruses has been developed, providing 97.8% identification accuracy.

The second part of the work is devoted to the development of innovative approaches to protecting Wi-Fi wireless networks. A conceptual model of the Wireless Honeypot as a Service information protection system using cloud computing is presented, which provides improved speed and deployment flexibility. A unique method for tracking attackers using metadata with 90–100% geolocation accuracy has been developed. A diagnostic model of a decoy system has been proposed, which allows configurations to be automatically generated according to the attacker's profile. A method for detecting intrusions based on the K-nearest neighbours algorithm has been presented, which provides 100% accuracy in detecting "evil twin" attacks.

The practical value of the monograph lies in the possibility of direct implementation of the developed methods and tools in cybersecurity systems. The research results can be used to protect both corporate and private networks. The proposed solutions significantly increase the level of protection against modern cyber threats, including ransomware and attacks on wireless networks.

The monograph will be useful for cybersecurity specialists, system administrators, software developers, researchers, teachers, and students of relevant specialities. The materials of the work are also of interest to managers of organisations and specialists responsible for the information security of enterprises of various forms of ownership.

KEYWORDS

Cybersecurity, ransomware, eBPF, artificial intelligence, machine learning, deep neural networks, wireless networks, IEEE 802.11, Wireless Honeypot as a Service, K-nearest neighbours (KNN), cryptographic ransomware, real-time, cloud computing, metadata, geolocation, information protection, system call monitoring, malware classification, evil twin attack, honeypot.

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 detecting and countering ransomware 4
1.1 Analysis of the current state of research and publications 4
1.2 Ransomware: description and classification, types 5
1.3 Basic models and methods for detecting and countering ransomware 10
1.4 Analytical review of existing approaches to securing computer networks from ransomware 15
1.5 Analysis of the current state of cybercrime in Ukraine and worldwide. The use of ransomware during the war in Ukraine 17

2 Using eBPF to detect ransomware 23
2.1 Introduction to eBPF: overview of architecture and capabilities 23
2.2 Methodology for developing eBPF-based modules 27
2.3 Using eBPF to track system calls 32
2.3.1 Combining Seccomp and eBPF policies into a single Seccomp-eBPF module 33
2.3.2 Seccomp notifier 34
2.4 Monitoring files and cryptographic activity using honeypots and eBPF 35
2.5 Network traffic analysis and process monitoring using eBPF 39
2.6 Using eBPF to monitor performance metrics 43
2.7 Accessing kernel data with eBPF 45
2.8 Implementation of eBPF modules on the Windows platform 47

3 Creating an integrated method for analysing ransomware based on machine learning models 49
3.1 Architecture of an integrated ransomware detection system 49
3.2 Development of a model pipeline 49
3.2.1 Formation of datasets 54
3.3 Choosing machine learning models for data analysis 55
3.3.1 Development of a model for classifying ransomware using decision trees and random forest ensembles 59
3.3.2 The support vector method as a tool for separating "safe" and "unsafe" programmes based on a hyperplane that maximises the distance between classes 62

3.3.3 Development of a model for detecting ransomware viruses using deep neural networks 67
3.4 Machine learning evaluation metrics 69

4 Analysis of the effectiveness of the proposed solutions 73
4.1 Organisation of a test environment for conducting ransomware attacks to evaluate the effectiveness of solutions 73
4.2 Conducting test attacks and collecting data for analysis 74
4.2.1 Safe simulation of various types of ransomware 75
4.3 Analysis of the results of machine learning models in various experiments 79
4.3.1 Accuracy, completeness, and other quality metrics 81
4.3.2 Performance consumed by system resources 84
4.4 Comparison of the effectiveness of different detection methods 85
4.5 Comparative analysis with traditional methods of detecting ransomware viruses 86

5 Status and prospects for the development of intrusion detection technology and decoy systems in IEEE 802.11 wireless networks 87
5.1 Features of existing intrusion detection and decoy systems for IEEE 802.11 networks 87
5.2 Approaches to organising the collection of information about attackers in IEEE 802.11 networks 90

6 Concept of building an information security system in IEEE 802.11 wireless networks using intrusion detection systems and-decoys 92
6.1 Development of an optimal life cycle for decoy systems for IEEE 802.11 standard networks 92
6.2 Monitoring and logging events 94
6.3 Researching possible unmasking signs in decoys 98
6.4 Development of a conceptual model of an information security system using intrusion detection systems and decoy systems 99

7 Development and optimisation of the information security model for IEEE 802.11 wireless networks.11 using intrusion detection systems and decoy systems 106
7.1. Development of a methodology for tracking attackers 106
7.2 Diagnostic model of a wireless network decoy system for the IEEE 802.11 standard 110
7.3 Detailed assessment of the complexity of bypassing the conditional protection of a decoy system with WPA/WPA2 wireless security protocols 115
7.4. Application of cloud computing to determine the security level of decoy systems and IEEE 802.11 wireless networks 119
7.5 Improving methods for detecting intrusions in IEEE 802.11 networks using artificial intelligence systems 122

7.5.1 Development of a method for identifying MAC address spoofing and "evil twin" attacks 124
7.5.2 Application of machine learning in studying the behaviour of IEEE 802.11 network users and subsequent intrusion detection 126
7.5.3 Using machine learning to detect intrusions in IEEE 802.11 networks 128

8 Improving mechanisms for identifying intrusions, the malicious person, and the effectiveness of decoy systems in IEEE 802.11 wireless networks 131
8.1 Research within the framework of developing a methodology for tracking attackers 131
8.2 Application of a diagnostic model of a decoy system for IEEE 802.11 wireless networks 135
8.3 Analysis of results on improving the methodology for distributed selection of access keys to the WPA2 protection mechanism in IEEE 802.11 networks 137
8.4 Detection of "evil twin" attacks on IEEE 802.11 (Wi-Fi) networks using the KNN classification model 140

Conclusions 153
References 155

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

The results presented in the monograph are of direct practical importance for organisations of various sizes, from small businesses to large corporations. The proposed methods and tools can be used to improve the level of protection of information systems against modern cyber threats, including ransomware and attacks on wireless networks. Particular attention is paid to the practical aspects of implementing the proposed solutions, including recommendations for integration with existing security systems and optimisation of settings for specific operating conditions.

The materials of the work will be useful for a wide range of specialists in the field of cybersecurity, including developers of protection systems, system administrators, researchers, and teachers of related disciplines. The monograph is also of interest to managers of organisations and specialists responsible for the information security of enterprises of various forms of ownership, as it provides a comprehensive understanding of modern approaches to cybersecurity and practical recommendations for their implementation.

INTRODUCTION

In today's digital world, cybersecurity is becoming critically important for society, business, and the state. The rapid development of information technology and its penetration into all areas of life not only creates new opportunities but also poses unprecedented challenges to the security of information systems. Cyber threats are constantly evolving, becoming more sophisticated and adaptive, which creates serious problems for traditional information security systems. Among the most pressing issues today, two critical areas stand out: countering ransomware and ensuring the security of wireless networks.

Ransomware has become one of the most serious threats to global cybersecurity, demonstrating an impressive ability to adapt and evolve. In recent years, there has been a rapid increase in the number and complexity of such attacks, leading to significant financial losses and disruption to critical infrastructure. The year 2020 was particularly telling, when numerous hospitals treating COVID-19 patients were hit by ransomware attacks, with serious consequences for the healthcare system. The economic damage from such attacks amounts to billions of dollars annually, with small and medium-sized enterprises, which often lack sufficient resources to restore their systems, being the most vulnerable.

The international nature of cyberspace makes it much more difficult to combat ransomware, as criminals often operate from territories where there are no effective mechanisms for legal cooperation between states. This makes it much more difficult to identify and prosecute them. Moreover, the emergence of cryptocurrencies has created a convenient mechanism for receiving ransoms, which further stimulates the development of this type of cybercrime.

At the same time, the widespread use of IEEE 802.11 (Wi-Fi) wireless networks has created a new vector for cyber threats. Despite constant improvements in security protocols, these networks remain vulnerable to various types of attacks. A particular danger is the possibility of attacks from a considerable distance — up to 2 km or more, which makes it significantly more difficult to detect and locate attackers. Traditional protection methods, such as signature analysis and static rules, are proving to be insufficiently effective against modern attack methods.

Wireless network security is becoming particularly critical in the context of the growing dependence of business and society on mobile technologies. The widespread adoption of the "bring your own device" (BYOD) concept in the corporate environment, the development of the Internet of Things (IoT), and the transition to hybrid working models create additional challenges for ensuring wireless network security.

The situation is complicated by the fact that modern cyber threats are highly adaptable and capable of evolving rapidly. Ransomware uses sophisticated evasion mechanisms, including polymorphism and metamorphism, which allow it to change its code with each execution. Zero-day attacks exploit unknown vulnerabilities for which no protection methods have yet been developed. An additional challenge is the growing activity of initial access brokers and insiders, who create additional threat vectors for information systems.

Targeted attacks (APTs — Advanced Persistent Threats) pose a particular danger, as they often use a combination of different techniques to achieve their goals. Such attacks can remain undetected for a long

time, gathering confidential information or preparing large-scale sabotage. Ransomware is increasingly becoming a tool for such complex, multi-stage attacks.

In this context, the development of new approaches to cybersecurity based on the use of artificial intelligence technologies is becoming particularly relevant. It is intelligent systems, capable of learning and adapting to new threats, that can provide the necessary level of protection in the constantly evolving landscape of cyber threats. Machine learning technologies and deep neural networks show significant potential in detecting complex patterns of malware behaviour and anomalies in network traffic.

This monograph presents a comprehensive approach to the creation of intelligent cyber defence systems, combining advanced technologies for detecting and countering ransomware with innovative methods of protecting wireless networks. The proposed solutions are based on the use of eBPF technology for monitoring system calls, the application of deep neural networks and ensembles of machine learning algorithms for analysing program behaviour and network traffic.

Considerable attention is paid to the development of methods that allow threats to be detected in real time or near real time (with a delay of no more than 30 seconds). This is especially important for preventing ransomware attacks, where the speed of the protection system's response can be a decisive factor in preventing data encryption and subsequent ransom demands.

The paper presents an innovative model of an integrated ransomware detection system that provides over 95% accuracy in detecting malicious programs. The developed methodology for applying deep neural networks to identify complex behaviour patterns of ransomware viruses demonstrates an accuracy of 97.8%, confirming the effectiveness of the proposed approach.

Particular attention is paid to the protection of wireless networks. The presented concept of Wireless Honeypot as a Service using cloud computing ensures the flexibility and scalability of the protection system. The developed method of tracking attackers by metadata achieves a geolocation accuracy of 90–100%, which significantly exceeds the performance of existing solutions. The proposed method of ing intrusion detection based on the K-nearest neighbours algorithm provides 100% accuracy in detecting "evil twin" attacks.

The work pays considerable attention to the problem of detecting and countering attacks on wireless networks using honeypots. The developed diagnostic model allows automatically generating honeypot configurations according to the profile of a potential attacker, which significantly increases the effectiveness of their use. The use of cloud technologies for deploying such systems provides the necessary flexibility and scalability of the solution.

An important aspect of the work is the development of methods for de-anonymising attackers in wireless networks. The proposed methodology uses a combination of different data sources, including device metadata and geolocation information, to create a comprehensive picture of the activity of potential attackers. This is especially important for countering targeted attacks and investigating security incidents.

The monograph is structured in such a way as to sequentially reveal all aspects of the creation and implementation of intelligent cyber defence systems. The first chapters are devoted to analysing the current state of the problem and the theoretical justification of the proposed approaches. Next, the developed methods and technologies are examined in detail, and the final chapters focus on the practical aspects of implementation and evaluation of the effectiveness of the proposed solutions.

The results presented in the monograph are of direct practical importance for organisations of various sizes, from small businesses to large corporations. The proposed methods and tools can be used to improve the level of protection of information systems against modern cyber threats, including ransomware and attacks on wireless networks. Particular attention is paid to the practical aspects of implementing the proposed solutions, including recommendations for integration with existing security systems and optimisation of settings for specific operating conditions.

The materials of the work will be useful for a wide range of specialists in the field of cybersecurity, including developers of protection systems, system administrators, researchers, and teachers of related disciplines. The monograph is also of interest to managers of organisations and specialists responsible for the information security of enterprises of various forms of ownership, as it provides a comprehensive understanding of modern approaches to cybersecurity and practical recommendations for their implementation.

1 ANALYSIS OF THE PROBLEM OF DETECTING AND COUNTERING RANSOMWARE

1.1 ANALYSIS OF THE CURRENT STATE OF RESEARCH AND PUBLICATIONS

Although the first ransomware virus was created in 1984 [1] and despite the fact that there are many methods of detection and counteraction, ransomware viruses are considered the greatest threat to cybersecurity today. Research into methods for detecting and countering ransomware includes both traditional signature- and behaviour-based methods and new approaches used for programme analysis, security information and event management (SIEM) systems, and network traffic adjustment. Based on a review of the literature, the following advantages and disadvantages of existing methods can be identified:

The article "Design and Implementation of an Intrusion Detection System by Using Extended BPF in the Linux Kernel" by S.-Y. Wang and J.-C. Chang is devoted to the development and implementation of an intrusion detection system using the extended Berkeley Packet Filter (eBPF) in the Linux kernel [2]. This research allows for the creation of more reliable and efficient intrusion detection systems compared to traditional approaches, which are based on user space and rely on tracking traffic patterns. However, this article has several areas for improvement, namely: the scalability of eBPF-based systems may be limited in large Windows-based environments, and the creation of cross-platform modules will address this need.

The article "Creating Complex Network Services with eBPF: Experience and Lessons Learned" by S. Miano et al. [3]. The article discusses the creation of complex network services using eBPF (extended Berkeley Packet Filter) technology. The authors share their experience and lessons learned during the development of complex network services based on eBPF. They also point out that creating complex network functions using eBPF has proven difficult due to the limited capabilities of the technology, which affects the flexibility of their application in real network environments. However, in 2018, Meta created the open-source Katran application for network traffic analysis.

The article "Demystifying the Performance of XDP BPF" by O. Hohlfeld, J. Krude, J. H. Reelfs, J. Ruth, K. Wehrle, examines the performance of eXpress Data Path (XDP) for network data analysis provided by the extended Berkeley Packet Filter (eBPF), focusing on factors that affect XDP performance [4].

The article "A Protocol-Independent Container Network Observability Analysis System Based on eBPF" by C. Liu et al. explores the development of a system for studying and analysing container network observability based on eBPF (extended Berkeley Packet Filter), which operates independently of network protocols [5].

The article "Detection of Denial of Service Attack in Cloud-Based Kubernetes Using eBPF" by A. Sadik et al. explores the use of the extended Berkeley Packet Filter (eBPF) to detect denial-of-service (DoS) attacks in clouds-based on Kubernetes. The article "Introducing SmartNICs in Server-Based Data Plane Processing: The DDoS Mitigation Use Case" by S. Milano et al. examines the implementation of SmartNICs in server processing for protection against DDoS (distributed denial-of-service) attacks [6, 7]. Both articles emphasise the need for innovative approaches to countering cyber threats, although they also highlight the need for further research to improve scalability and integration with other security solutions.

The article "A Framework for eBPF-Based Network Functions in an Era of Microservices" by S. Miano discusses a new framework for implementing network functions based on eBPF (extended BPF) in the context of microservices, facilitating the development and deployment of software network functions based on eBPF [8].

Improving methods for detecting and countering ransomware viruses in real time is an important issue in the field of cybersecurity. The use of eBPF can provide significant advantages and help overcome certain shortcomings in researching this issue.

Considering the articles mentioned, the following research advantages can be highlighted on the topic of eBPF:

1. High processing speed: eBPF allows for much faster processing of network traffic and comprehensive real-time activity monitoring compared to more traditional counterparts operating at the user space level.

2. Higher detection accuracy: eBPF allows for the development of flexible and adaptive detection systems, which helps to more accurately detect unauthorised activity in the early stages of an attack.

3. Flexibility: eBPF allows virus detection and countermeasures to be integrated directly into the operating system kernel, enabling deeper analysis of network traffic and rapid adaptation to the latest types of attacks.

4. BPF promotes the automation of security processes, enabling immediate detection and response to threats using integrated real-time solutions.

Disadvantages of research:

1. Development complexity: using eBPF to develop virus detection and countermeasure systems can be a complex process that requires in-depth knowledge of eBPF and network security. Close collaboration between cybersecurity development teams is necessary to ensure successful implementation.

2. Hardware limitations: effective implementation of eBPF may depend on the availability of modern hardware, including network adapters, which can still be expensive or difficult to acquire and deploy.

3. Insufficient research in specialised and specific contexts: in the context of research, the increased use of eBPF is a relatively new area of research, which may require further research to implement and evaluate its effectiveness in different contexts and environments.

Based on these advantages and disadvantages, it can be concluded that eBPF has significant potential for real-time detection and countering of ransomware. However, in order to achieve the best results for a variety of scenarios and environments, cybersecurity developers need to make a concerted effort to develop and research effective eBPF-based techniques and solutions.

1.2 RANSOMWARE: DESCRIPTION AND CLASSIFICATION, TYPES

In today's world, where information technology has become an indispensable part of everyday life, information security is a key challenge for organisations in all areas of activity. One of the most malicious and threatening types of cyber attacks is ransomware, also known as ransomware [9, 10].

Ransomware is malicious software that encrypts user data and demands compensation for decryption. It can lead to significant financial losses, business interruptions and reputational damage, especially in

a business environment. Ransomware can be a major problem not only for individual users, but also for large organisations. In terms of distribution strategies, ransomware viruses are often spread among users, which can be divided into those spread through spam campaigns, fraudulent websites, infected applications, or exploitation of system vulnerabilities [11]. Depending on their impact on the infected system, they can be divided into encryption ransomware, which encrypts user data, and blocking ransomware, which blocks access to the system as a whole. There are also many other subtypes of ransomware, including fairly new forms such as ransomware for mobile devices, "insider" ransomware that infects and encrypts data already inside the network, and so-called "destructive" ransomware viruses, which do not actually provide the ability to recover data, even if the ransom has been paid [12–24].

Firstly, ransomware infection most often occurs when a user visits a website with security breaches. A popular method used by attackers is to send a phishing email containing a link to a website hosting the ransomware code. This form of attack, also known as "social engineering" [25–27], does everything possible to appear legitimate.

Secondly, ransomware code is designed to infect systems through one of many known software or operating system vulnerabilities. Additional forms of ransomware infection specifically target users with higher privilege levels, such as administrators, to introduce malicious code.

Once the code has been delivered and launched on the system, two things can happen. The encryption program will block users from accessing the system. Cryptographic ransomware programs encrypt data using complex mathematical encryption keys. Systems affected by a ransomware attack may suffer significant damage, or certain types of files or systems may be targeted, such as SQL databases or Microsoft Office files.

Types. Ransomware is a type of malware and inherits many techniques from other malicious programs. Most techniques are explained in the context of Windows, so some Windows APIs have been mentioned. There are the following types of ransomware [28]:

1. Scareware and fake security software.
2. ScreenLocker.
3. Browser ransomware.
4. Crypto-ransomware.
5. Ransomware targeting infrastructure.
6. Boot ransomware.

For each type of ransomware, the following points are considered:

– techniques used with the ransomware family;

– some popular ransomware in the family;

– guidelines for analysing such ransomware for malware analysts.

ScreenLocker Ransomware. This ransomware does not encrypt files on the victim's machine. It locks the entire screen and prevents the victim from doing anything else until they pay the ransom. Here is a list of some popular ScreenLocker ransomware:

1. Reveton.
2. Urausy.
3. Kovter.

4. Tobfy.
5. Weelsof.
6. BlueScreen.
7. Koktrom.
8. Winlock.
9. LockScreen.

Boot Ransomware. There are several types of malware that can run during system startup. They are sometimes referred to as bootkits. Boot ransomware is a bootkit that displays a warning message before the user logs into the operating system [29].

Boot programs take control before the operating system is fully loaded (**Fig. 1.1**). As a result, the victim is left with few options. They also complicate the work of researchers. Petya is a ransomware virus that infects the master boot record (MBR) [30].

Before we move on to infecting the boot file, let's briefly review the Windows boot process. In order for a computer to boot successfully, all of these components must be working properly: the BIOS, the operating system, and the hardware. If one of these components fails, it will most likely cause the boot sequence to fail. When the processor is turned on, the BIOS boots from the BIOS ROM. The BIOS initiates POST (power-on self-test), which checks that devices such as the keyboard, RAM, and drives are working properly. The BIOS searches for a boot device. The MBR is the beginning of the first partition of the disk, or you could say that it is present in sector 0 of the physical hard disk. The MBR is read into memory and executed, and it starts with a code called the BootStrap loader. The MBR has a table called the partition table (pt) that stores partition information. The partition table has only one active partition, which is called the boot partition. The first sector of the active partition is called the boot sector or Volume Boot Record (VBR). The VBR is one of the most important structures and can contain the block size, partition size, MF, etc. The Master File Table (MFT) is a table that contains information about files, their size, timestamp (when they were created or modified), file access (read/write permissions), and so on. The MFT is present in the NTFS file system in Windows. Petya is known to encrypt the MFT [29]. This way, Windows will not know the location of the files. Even if individual files are not encrypted, they cannot be recovered because the MFT, which is the knowledge base for the files, is encrypted.

The BootStrap loader loads the sector into memory and transfers control to it. The VBR finds and loads the loader code. In Windows XP, the boot loader is NTLDR, and in Windows 7, the Boot Configuration Database (BCD) is used. In Windows XP, NTLDR finds a list of operating systems to boot. It is located in the boot.ini file. NTLDR loads the registry and devices needed during boot. In Windows 7, BOOTMGR is used instead of NTLDR, and the list of operating systems to boot is contained in the Boot Configuration Database (BCD). After this stage, winload.exe loads the registry and devices in Windows 7. Control is then transferred to NTOSKRNL.exe, which loads the drivers and services required by the system:

Crypto Ransomware. Ransomware viruses are indeed one of the most harmful types of malware today. They encrypt files on the user's computer, effectively blocking access to their own data. The ransomware then demands a ransom in exchange for a decryption key to restore access to the files [25, 26, 30]. One possible reason for the increase in cryptocurrency ransomware may be the relative ease of development

compared to other forms of malware. Instead of writing complex code to infiltrate a system or steal data, ransomware simply scans the user's directories for relevant files — usually those that appear to be personal or important — and encrypts them. Many types of ransomware viruses do not bother to hide in the system or install rootkit components, as they only need to run once to encrypt all files. Some of them also have mechanisms to check whether the system is infected with other encryption viruses to avoid duplication. There are many types of encryption programs. Here are some of the most notable examples: Locky, Cerber, CryptoLocker, Petya.

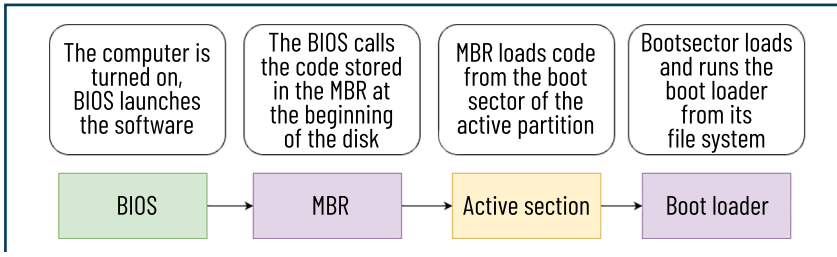

Fig. 1.1 Windows boot process

How it works. Ransomware viruses technically perform the following actions:

1. Find files on the local system. On a Windows computer, it can use the FindFirstFile() and FindNextFile() APIs to search through file directories. Many ransomware programs also search for files on shared drives.

2. Next, it checks the file extension that needs to be encrypted. Most of them have a clearly defined list of file extensions that need to be encrypted. Even if it encrypts executable files, it should not encrypt any system executable files. This ensures that it will not be possible to restore files from a backup by deleting the backup. Sometimes this is done using the vssadmin tool [31]. Many ransomware viruses use the vssadmin command provided by Windows to delete shadow copies. Shadow copies are backup copies of files and volumes. The vssadmin tool (vss administration) is used to manage shadow copies. The abbreviation VSS stands for Volume Shadow Copy, also known as the Volume Shadow Copy Service. **Fig. 1.2** below shows a screenshot of the vssadmin tool.

After encrypting the files, the ransomware leaves a text file for the victim. It usually states that the files on the system have been encrypted and that a ransom must be paid to decrypt them. The ransom note contains instructions on how to pay the ransom. Today, many cryptographic algorithms are used by malicious programs. Malicious software can use cryptography for the following purposes:

— to obfuscate its own code so that antivirus software cannot easily identify the real code;
— to communicate with its own server;
— to encrypt files on the victim's computer.

A cryptographic system may have the following components:

— plaintext;
— encryption key;

– ciphertext, i.e. encrypted text;
– encryption algorithm, also called a cipher;
– decryption algorithm.

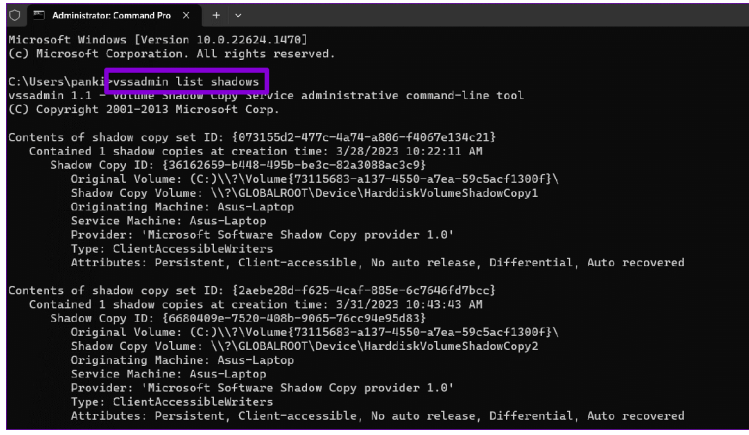

Fig. 1.2 Vssadmin utility

There are two types of cryptographic algorithms depending on the type of key used (**Fig. 1.3**):
– symmetric;
– asymmetric.

Explanation of the algorithm: the sender is the person who sends the data after it has been encrypted, and the recipient is the person who decrypts the data using the key. In symmetric key encryption, the sender and recipient use the same key, also known as a secret key. The sender uses the key to encrypt the data, while the recipient uses the same key to decrypt it. The following algorithms use a symmetric key:
– RC4;
– AES;
– DES;
– 3DES;
– BlowFish.

A symmetric key is easier to implement, but it faces the problem of secure key exchange [32–36]. An open or asymmetric key solves the key exchange problem by using a pair of keys: public and private. The public key can be distributed unprotected, while the private key is always kept secret by the owner. Either key can be used for encryption, and the other for decryption.

Here are the most popular asymmetric encryption algorithms:
– RSA;

– Diffie-Hellman;
– ECC;
– DSA.

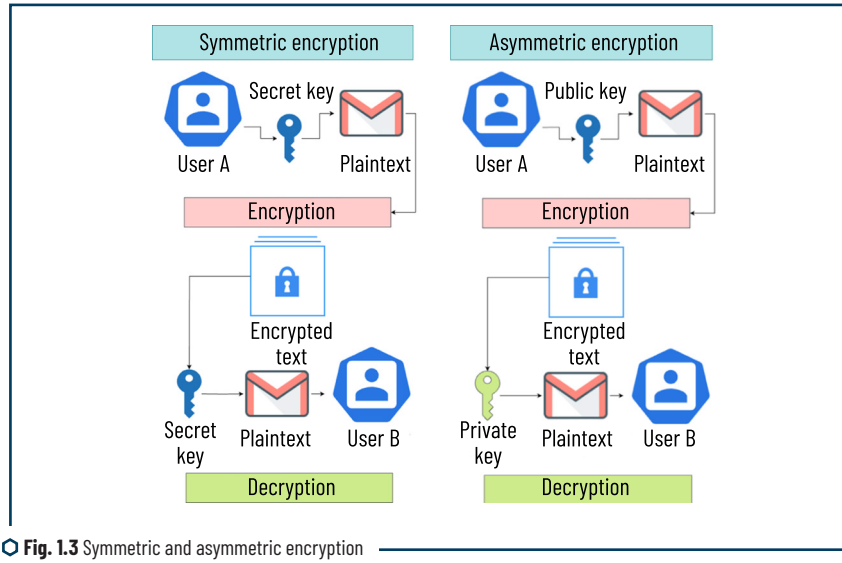

Fig. 1.3 Symmetric and asymmetric encryption

Cryptographic ransomware began with simple symmetric key cryptography. But soon, researchers were able to easily decrypt these keys. Therefore, they began to use asymmetric encryption. The current generation of ransomware programmes has begun to intelligently use both symmetric and asymmetric keys. The tools and concepts of malicious programmes remain the same [37–41].

1.3 BASIC MODELS AND METHODS FOR DETECTING AND COUNTERING RANSOMWARE

The main models and methods for detecting and countering ransomware in modern computer security are static analysis, dynamic analysis, and proactive detection methods.

Static analysis. Focuses on studying the program code of viruses without executing them. This approach includes analysing hashes, strings, or using machine learning to classify malicious code. However, it may be ineffective against viruses that use code obfuscation techniques. Static analysis involves determining the characteristics of a file, such as its type and other features that can be identified without running it [42].

Antivirus researchers collect many variants of the same family of malicious software, identify common static properties in them, and create a digital signature. The static properties used in the signature may

include the hash of certain areas of the file, properties, size, etc. But since strains often vary statically, antivirus products must regularly update their signatures.

Dynamic analysis is a method that involves running viruses in a controlled environment, such as a sandbox, and observing their behaviour, such as system resource usage, network activity, and system changes. Unlike static analysis, dynamic analysis can detect viruses that use code obfuscation, but it is more resource-intensive and time-consuming [43].

Dynamic analysis is one of the most important stages of malware analysis. In addition, dynamic analysis can be called behavioural analysis, since this type of analysis describes what malicious code does, in other words, the behaviour of an .exe file, or the changes that occurred in the system after running this file.

Each of these methods has its advantages and disadvantages, and none of them can guarantee 100% protection against ransomware (**Table 1.1**). For these reasons, to solve the problem of detecting and countering ransomware, it was decided to use eBPF technology, which allows tracking system calls at the operating system kernel level, thereby providing in-depth analysis of system processes.

Table 1.1 Comparison of types of virus analysis

Type of analysis	Advantages	Disadvantages
Static analysis	1. Speed: static analysis can be performed quickly because it does not require the virus to be executed. 2. Safety: it poses no threat because the programme is not run	1. Obfuscation: viruses that use obfuscation or polymorphism can evade detection. 2. Lack of context: static analysis does not provide information about how the programme will perform in a real environment
Dynamic analysis	1. Detailed analysis: provides more information about program behaviour, including resource usage, network activity, and system changes. 2. Effectiveness against obfuscation: can detect viruses that use code obfuscation techniques	1. Time consumption: dynamic analysis usually takes more time than static analysis. 2. Potential threat: although the analysis takes place in a controlled environment, there is a risk that the virus could escape these restrictions
Proactive detection methods	1. Predicting new threats: can detect unknown threats based on the characteristics of known viruses. 2. Adaptability: machine learning models can adapt to changes in attacks	1. False positives: proactive methods can sometimes mistakenly identify legitimate programmes as malicious. 2. Training requirements: machine learning models require a large data set for training, which can be challenging
Countermeasures	1. Real-time protection: countermeasures can block malicious activity in real time, preventing potential damage. 2. Isolation: techniques such as virtualisation and containerisation can isolate the impact of ransomware on the system	1. No 100% protection: there is no guarantee that countermeasures will always be able to repel an attack. 2. Vulnerability risk: security relies on trust in the solutions used, which may also have vulnerabilities

We can look for the following changes in the Windows system to identify malicious software:

— changes to the file system;

— registry changes;

– network connection;

– process changes.

Changes to files, the registry, and the network are straightforward. Process changes can include changes in the creation of new processes and threads. Process changes also include changes in the virtual memory of a process. There are tools available to monitor the changes mentioned above. A set of virtual machines for analysing malware should have at least one tool that falls into the following categories:

– file monitoring tool: filemon, procmon;

– registry monitoring tool: regmon, procmon;

– autoruns tool;

– network monitoring tool: Wireshark, fakenet, Microsoft Network Monitor;

– API logger: StraceNT, sandboxes, Sysnalyzer API logger;

– process inspection tool: Process Explorer, Process Hacker;

– sandboxes can be considered equivalent to a combination of these tools.

Monitoring files and registries. File and registry changes are one of the important events used to identify malware. Microsoft Sysinternals provides regmon and filemon for this purpose. Sysinternal offers procmon, which can cover registry and file monitoring.

Fig. 1.4 shows the activity of reading (ReadFile), closing (CloseFile) and writing (WriteFile) a file by the vssvc.exe process. vssvc.exe creates the file C:\Users\amohanta\Favorites\Microsoft Websites\Microsoft Store.url.readme_txt, then writes to it and closes it after writing.

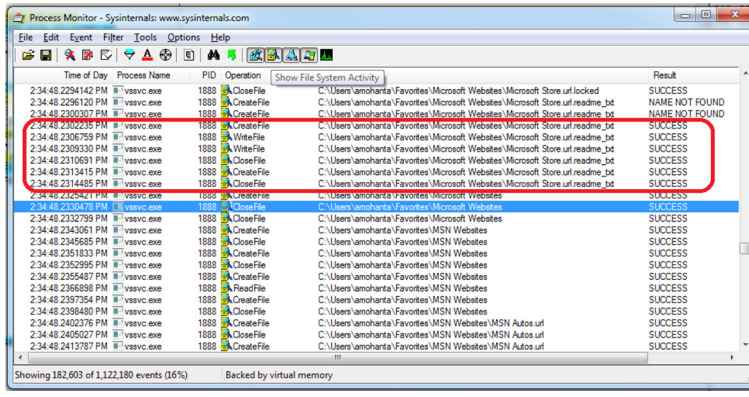

Fig. 1.4 Procmon demonstrates the file activity of the vssvc.exe process

Procmon can also monitor process activities such as thread entry and exit, network connections, and more. It is important to filter activities because many system processes constantly make changes to files and the registry.

Microsoft Sysinternal Autoruns (**Fig. 1.5**) is a useful tool for identifying all processes that are automatically activated when Windows starts.

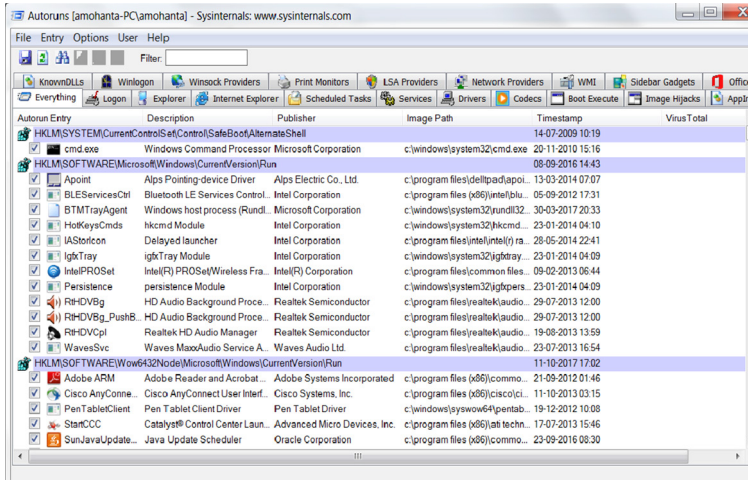

Fig. 1.5 Autoruns utility

This tool displays a lot of information. It includes entries related to startup, scheduled tasks, and services that can activate malicious programs when Windows starts. This tool can also be used for troubleshooting and forensic analysis to identify unwanted software that may be running without the user's knowledge.

Most network professionals are familiar with the Wireshark tool. Wireshark is available on both Linux and Windows. Microsoft provides the Microsoft Network Monitor tool, which has the ability to capture packets. One of the added benefits is that it shows us which process is generating the network communications. It is easy to associate a network connection with a process, which is an additional advantage over other network monitoring tools (**Fig. 1.6**).

This simplifies the analyst's work by allowing them to determine which process is creating the connection. When unusual network connections are detected from Windows system processes such as explorer.exe and winlogon.exe, there may be suspicion that malicious code has been injected into the process.

API logging tools show the sequence in which an executable file calls the API. The record for a specific API may include the parameters passed to the API and the values returned by it. StraceNT and apimonitor are some of the tools that can be used for API logging.

The tool has a user-friendly interface and advanced functionality that facilitates analysis. Logging API events is critical for interpreting activities related to changes in software system processes.

You can configure the tool by going to the Capture menu. You will be able to change the API Filter value. You can then select the APIs you want to log, such as Registry. You can start logging again by going to the Capture menu and clicking Capture API Events (**Fig. 1.7**).

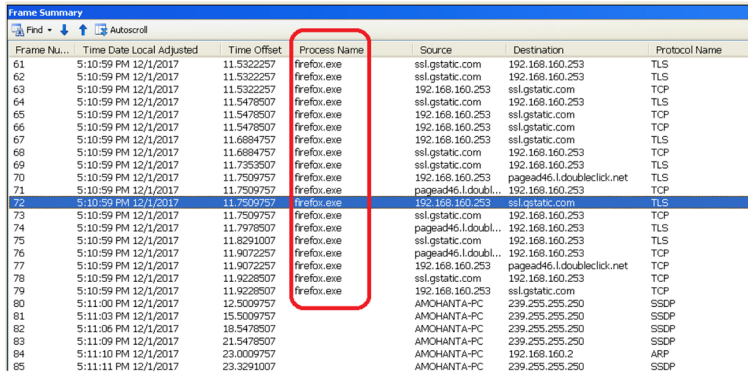

Fig. 1.6 Network monitoring

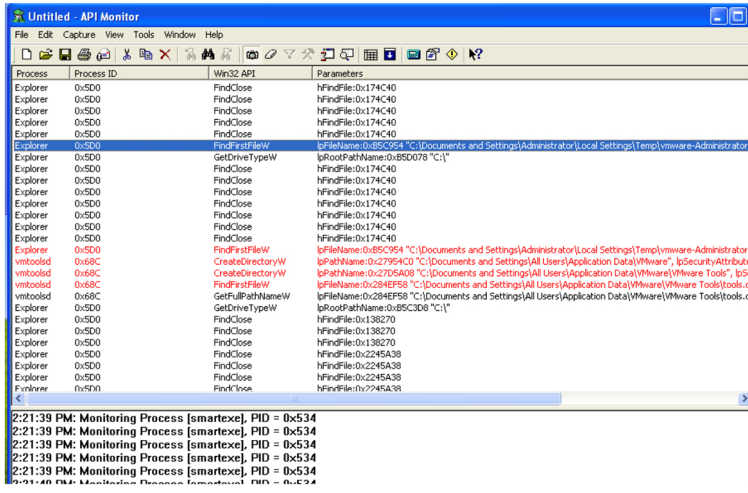

Fig. 1.7 Apimonitor logs

Malware analysts can use API logs to find out how malware works internally.

Process inspection is a powerful method. The strings present can be checked in the virtual memory of the process. When analysing strings in a file on disk, nothing meaningful is usually found (static analysis using a string tool), as the malware remains hidden. When packed malware is executed, a process is created in which the malware unpacks the code necessary for execution and malicious activity in virtual memory.

Therefore, it can be said that the actual strings related to the malware are observed in virtual memory.

Process Explorer and Process Hacker are used to view strings in the virtual memory of a process.

The image below, **Fig. 1.8**, shows a snapshot of the Windows calculator process, calc.exe, in the Process Explorer tool. The program window opens when you click on the process entry in Process Explorer. Next, you can go to the Strings tab and click on the Memory button to view the strings stored in the virtual memory of the process. Clicking the Image button displays the strings found in the calc.exe file on disk, similar to what the strings tool used for static analysis does. Process Hacker is a similar tool, but provides additional details.

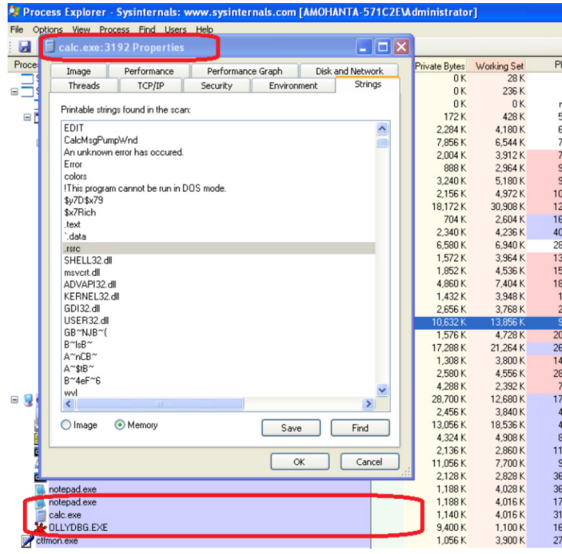

Fig. 1.8 Strings in virtual memory for the calc.exe process

1.4 ANALYTICAL REVIEW OF EXISTING APPROACHES TO SECURING COMPUTER NETWORKS FROM RANSOMWARE

The scientific community is gradually moving away from traditional network-oriented security (**Fig. 1.9**) in favour of an approach based on process behaviour analysis. This allows for control and decision-making

based not on packet header analysis, but on a detailed understanding of process dynamics. In reality, it is difficult to find examples of recent complex attacks based on packet interception [44].

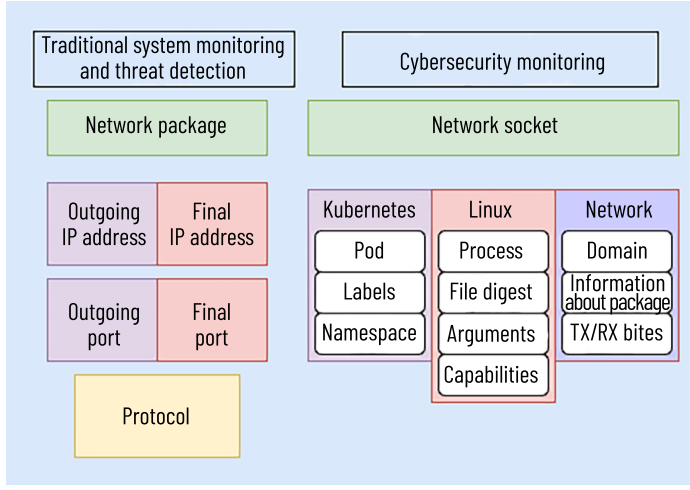

Fig. 1.9 Traditional network monitoring and threat detection. Cybersecurity monitoring

Computer network security is a fundamental issue that requires constant attention and updating, as cybersecurity threats are constantly evolving. Detecting and countering ransomware requires specialised tools and approaches. Modern methods include the use of network monitoring, intrusion detection systems (IDS), intrusion prevention systems (IPS), endpoint protection systems (EPS) and other technologies [45].

However, not all of these systems are effective against ransomware. For example, traditional antivirus programs typically rely on signature-based threat detection, which means they can only detect known variants of ransomware. However, given the rapid development and constant improvement of ransomware, this approach may be insufficient (**Fig. 1.10**).

Intrusion detection and intrusion prevention systems operate at the network level and detect abnormal behaviour that may indicate an attack [46]. However, these systems may also be insufficiently effective against ransomware, especially in cases where ransomware uses masking techniques to avoid detection.

The challenge is to develop more advanced approaches to detecting and countering ransomware that can adapt to new threats and use more sophisticated detection methods. One of the key areas developing in this direction is the use of eBPF (Extended Berkeley Packet Filter), which allows for the creation of more flexible and dynamic behaviour-based threat detection rules.

Further in this section, we will examine the architecture of SIEM and EDR solutions, analyse the current state of cybercrime in Ukraine and worldwide, justify the choice of research direction, and set the research objectives.

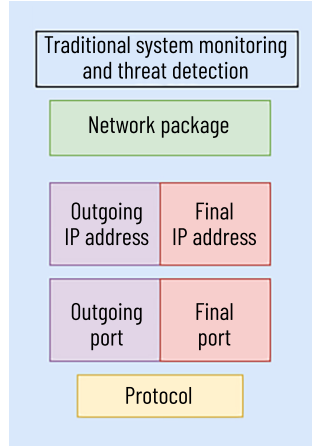

Fig. 1.10 Traditional network monitoring and threat detection

Security Information and Event Management (SIEM) and Endpoint Detection and Response (EDR) systems are key tools in modern cybersecurity. However, their effectiveness against ransomware can often be limited due to the use of traditional signature- and anomaly-based threat detection methods. The SIEM architecture is based on the centralised collection and analysis of security events from various sources on the network. SIEM systems are capable of aggregating and correlating large amounts of data, which helps to detect complex threats and develop responses to them [47]. However, SIEM systems may be less effective against new and unknown ransomware viruses that do not match known threat signatures.

EDR systems, on the other hand, focus on protecting endpoints — user devices such as computers and servers. They use a more advanced set of tools to detect threats, including behaviour monitoring, memory analysis and network analysis [48]. However, despite this, EDR systems can also be limited in their ability to detect and block ransomware, especially those that use techniques to bypass antivirus protection and detection. It should be noted that SIEM and EDR architectures are not mutually exclusive, and ideally both systems should be used together for the most complete protection. However, given the constant and evolving challenges in the field of cybersecurity, there is a need to develop new strategies and implement technologies, such as eBPF, that can complement and improve the application of SIEM and EDR systems.

1.5 ANALYSIS OF THE CURRENT STATE OF CYBERCRIME IN UKRAINE AND WORLDWIDE. THE USE OF RANSOMWARE DURING THE WAR IN UKRAINE

This list is constantly changing due to various factors. As of September 2023, some of the key groups active in the field included:

1. REvil (Sodinokibi): one of the most active groups using malicious software.

2. Conti: this group is known for its sophisticated attacks and high ransom demands.

3. DarkSide: this group gained notoriety after launching an attack on Colonial Pipeline, which caused significant fuel supply disruptions in the United States in May 2021.

4. LockBit 2.0: a strain of malware-as-a-service (RaaS) that follows the trend of double extortion, i.e. they encrypt victims' files and threaten to publish the stolen data.

5. Avaddon: this group was one of the most aggressive, demanding ransom and threatening to leak data if their demands were not met.

6. Ryuk: despite being in the malware scene for a long time, the Ryuk group continues to pose a significant threat, especially to the healthcare sector.

7. DoppelPaymer: known for its high ransom demands and attacks on large organisations.

These groups pose serious threats due to their sophistication and scale of operations. It is also worth noting that malware attacks are cyclical in nature and often come in waves, so periods of relative calm can often be followed by a flurry of attacks.

Global trends.

Fig. 1.11 shows that, as of today, Chainalysis has detected payments from ransomware programmes in 2021 amounting to just over 602 million USD. However, as was calculated last year, this figure is underestimated, and the actual total amount for 2021 is likely to be much higher. In fact, despite these figures, individual reports, as well as the fact that ransomware revenues in the first half of 2021 exceeded those in the first half of 2020, suggest that 2021 will ultimately be an even bigger year for ransomware.

The latest data for the first half of 2023 indicates that malicious activity is on track to break previous records, as both large and small payments are on the rise.

According to a report by Chainalysis, a company that analyses blockchain technology, ransomware is the only category of cryptocurrency crime showing growth this year, while all others, including hacks, fraud, malware, sales of abuse materials, scam shops, and darknet market revenue, are showing a sharp decline (**Fig. 1.12**).

Ransomware is the only form of cryptocurrency crime that is growing in 2023, according to the Chainalysis report. In fact, attackers using ransomware have reached the second-highest figure in history, demanding at least 449.1 million USD by June.

As shown in the Chainalysis chart below (**Fig. 1.13**), cumulative annual ransomware revenue for 2023 reached 90% of the total for 2022 in the first half of the year.

Comparative analysis of data from 2022 and 2023. If the growth rate of revenue remains at this level, in 2023 attackers will receive approximately 900 million USD from victims, which is slightly below the record figure for 2021 — 940 million USD. Analysts believe that the driving force behind this sharp increase in revenue is the so-called "big game hunting", as cybercriminals have returned to targeting large organisations from which they can extort large sums of money. This is reflected in the ransom payment distribution chart, which shows unprecedented growth in the right-hand side of the chart for the first half of 2023, corresponding to large payments. BlackBasta, LockBit, ALPHV/Blackcat and Clop lead as the main recipients of large payments, with Clop's average payment amounting to 1.7 million USD and the median to 1.9 million USD.

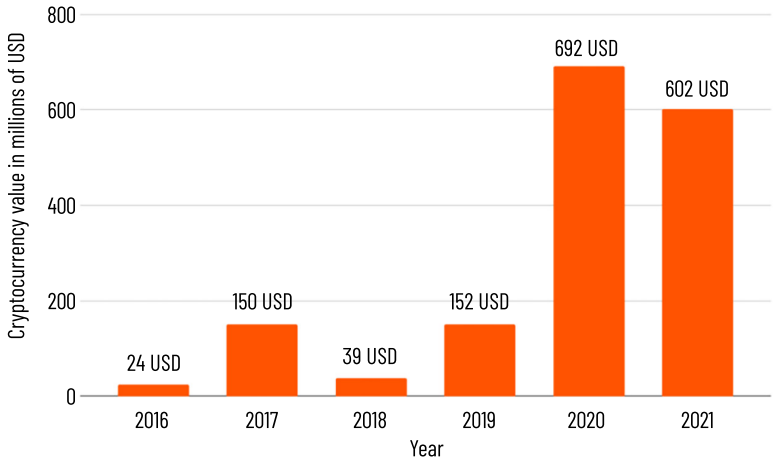

Fig. 1.11 Chainalysis shows the total amount of cryptocurrency obtained through ransomware viruses

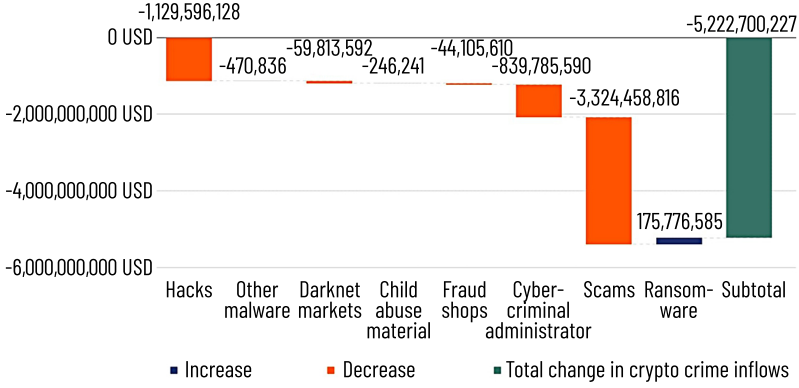

Fig. 1.12 Trends in illicit income – comparison of 2022 and 2023

Clop is responsible for two massive waves of attacks that exploited two zero-day vulnerabilities in file transfer tools: Fortra's GoAnywhere in the first quarter of the year and Progress's MOVEit Transfer in the second.

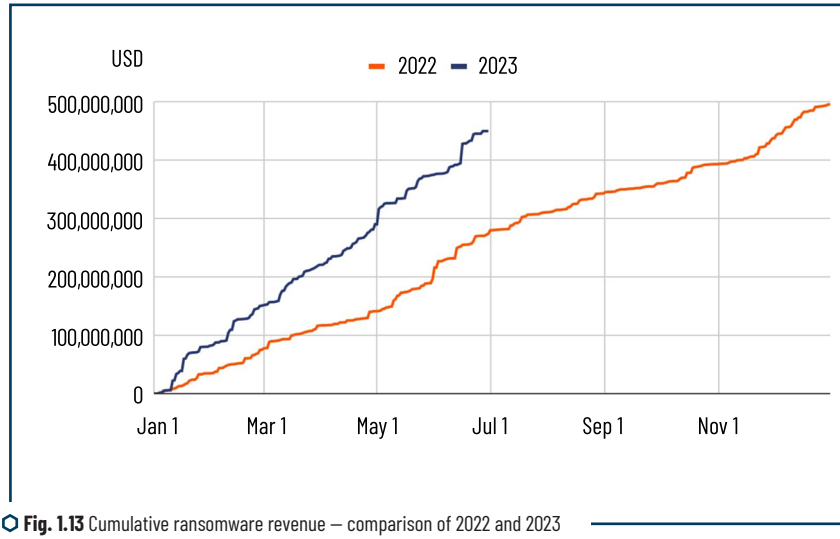

Fig. 1.13 Cumulative ransomware revenue — comparison of 2022 and 2023

In fact, Clop's GoAnywhere campaign, which included 129 attacks, made March 2023 a record month, as reported by the NCC Group at the time. The wave of MOVEit attacks is already more extensive, with 267 victims, and more information is appearing on the Clop ransomware website every week. The upward trend in the first half of 2023 is also evident at the other end of the spectrum: relatively small ransom payments targeting a wide range of victims, including individuals and small businesses, such as Dharma, Phobos and STOP/DJVU. The cybercrime ecosystem continues to evolve rapidly. One of the defining factors of the ransomware landscape in 2022–2023 is the ongoing war in Ukraine, which has a particular impact given the high concentration of cybercriminal groups from Russia and its neighbours. In the first months after the invasion, the scale and direction of ransomware and other cybercrimes changed, indicating a shift in priorities and constraints for actors operating in this space. Although the conflict has not led to the full-scale cyberwar that some commentators predicted, cyberattacks against Ukraine have been constant and significant both before and during the invasion, and have also been an active part of Ukraine's own defensive operations (**Table 1.2**).

One possible side effect of the war is the reorientation of cybercriminals from Russia and neighbouring countries in two directions. First, there is speculation that some criminals have shifted their focus from lucrative cybercrimes, such as ransomware attacks, to participating in military operations. In addition, despite the fact that ransomware attacks typically occur during wartime, they continue unabated in Ukraine.

Furthermore, despite the fact that ransomware activity from Russia and its neighbouring countries continues, these cybercriminals also appear to be expanding their targets, paying more attention to the Global South, including refocusing on targets in Asia and Latin America and moving away from critical infrastructure and other sensitive targets in NATO countries. This shift away from critical infrastructure and

other sensitive targets in NATO countries may be driven by a desire to avoid incidents that could exacerbate tensions between Russia and NATO member states.

Table 1.2 Number of payments

Name of ransomware strain	Average payment for 2023, USD	Median payment amount for 2023, USD
Dharma	265	275
Phobos	1,719	300
Stop/djvu	619	563
BlackBasta	—	147,106
ALPHV/Blackcat	1,504,479	305,585
Clop	1,730,486	1,946,335

The long-term implications of the invasion for the cybersecurity ecosystem remain uncertain. Several unresolved questions include how this conflict affects safe havens for cybercriminals and what the cybercrime ecosystem will look like as the conflict between Ukraine and Russia evolves. Furthermore, as the conflict unfolds, additional reports are blurring our understanding of the overall direction of ransomware. As noted earlier, the picture of the situation remains incomplete [48–52].

Initial access brokers. According to Recorded Future estimates, there were 65,000 ransomware attacks in 2020. That is simply too many victims for even a massive network of participants and their affiliates to access, steal files, and deploy ransomware infrastructure. The significance lies in scanning the Internet to identify vulnerable systems. Some initial access brokers (IABs) specialise in using stolen credentials, where an attacker attempts to log in using common username/password combinations using a brute force method in a rapid manner, while others focus on reusing credentials, where the attacker finds username/password combinations on underground markets and attempts to use them on targets. The role of IABs in ransomware attacks is to gain and maintain an initial foothold. They then sell access to ransomware actors for an average price of 5,400 USD. Many IABs specialise in exploiting other vulnerable systems, such as:

— Pulse Secure VPN;
— Citrix;
— Fortinet VPN;
— SonicWall Secure Mobile Access;
— Palo Alto VPN;
— F5 VPN.

Money laundering is difficult for ransomware groups. Ransomware actors have moved from conducting a few simple transactions that hide their money to laundering millions of dollars in collected ransoms. When money launderers from the Clop ransomware group were arrested in June 2021, it was reported that they had successfully laundered more than 500 million USD in collected ransoms.

Ransomware operators transfer most of the funds stolen from their victims to major exchanges, high-risk exchanges. Several RaaS operators emphasise that their payment portal is integrated with mixing services as a feature to attract affiliate talent. 73% of all funds controlled by ransomware actors were sent to just 83 deposit addresses by June 2021.

In 2020, Chainalysis Inc. identified 100 OTC brokers specialising in transferring money for cybercriminals [31]. OTC brokers are individuals or companies that hold large amounts of cryptocurrency. When a trader wants to exchange cryptocurrency for another type of cryptocurrency or fiat currency anonymously, they can agree on a fixed price with an OTC broker, who then executes the transaction.

Exploit brokers. Researchers have long noted that attackers associated with ransomware are actively buying access to exploits. This practice really came to light in the case of the Kaseya REvil ransomware attack [53]. During this attack, REvil, or one of its partners, exploited a previously unknown vulnerability (commonly referred to as a "zero-day" vulnerability) against Kaseya's Virtual System Administrator (VSA) software. Kaseya VSA is remote management software often used by managed service providers (MSPs) to remotely administer and secure their customers, especially smaller customers with limited IT or security staff [53]. In this case, the Kaseya network was never compromised — the REvil affiliate exploited the vulnerability to target MSPs that used Kaseya's VSA tool. Even then, the affiliate did not encrypt the MSP networks, but used its access to deploy ransomware to the MSP's customers.

At the time of writing, the attack on Kaseya VSA is the most well-known use of an exploit by a ransomware cybercriminal group. But ransomware groups regularly combine exploits as part of their attack strategy. They typically target known vulnerabilities for exploitation rather than zero-day vulnerabilities. Known exploits still work because ransomware groups and IABs rely on the slow patch cycle that many organisations maintain.

In book "This Is How I Tell the End of the World" journalist N. Perlroth details the growth of the exploit market and the competition between nation-state actors to obtain and exploit zero-day vulnerabilities. Due to the large sums of money that ransomware groups have earned over the past few years, especially with the advent of RaaS, they are able to compete with many nation-state actors for exploits.

2 USING EBPF TO DETECT RANSOMWARE

2.1 INTRODUCTION TO EBPF: OVERVIEW OF ARCHITECTURE AND CAPABILITIES

Berkeley Packet Filter is a Linux kernel component that allows user code to be executed on a built-in virtual machine in the kernel. This system is divided into two main parts: traditional BPF (cBPF) and enhanced BPF (eBPF, or simply BPF). The older cBPF version was limited to packet analysis, while the modern eBPF offers much broader capabilities, including packet modification, system call argument adaptation, user space program modification, and more.

The history of Berkeley Packet Filter (BPF) began in 1992, when it was introduced as an effective method for intercepting network packets for user monitoring. This mechanism was first documented by S. McKenna and V. Jacobs of Lawrence Berkeley National Laboratory. They described the BPF programming system using a unique set of 32-bit instructions similar to assembly language. Below is an example of code taken directly from their publication:

```
ldh    [12]
  jeq #ETHERTYPE IP, L1, L2
  L1:    ret #TRUE
  L2:    ret #0
```

This code snippet filters out packets that are not Internet Protocol packets. An Ethernet packet is fed into this filter, and the first command (ldh) loads a 2-byte value starting at byte 12 of that packet. In the next instruction (jeq), this value is compared to the value representing the IP packet. If they match, execution proceeds to the instruction labeled L1, and the packet is accepted, returning a non-zero value (labeled #TRUE here). If the values do not match, the packet is not an IP packet and is rejected with a return value of 0.

BPF has become an effective way to change kernel behaviour thanks to the following key features:

— it operates as a separate virtual machine, specifically optimised for interaction with register processors;

— BPF programmes use buffers for each individual application to avoid the need to copy packet information for processing;

— to ensure safety and prevent infinite loops, BPF programs are verified before being loaded into the kernel space.

eBPF, which uses 64-bit registers, is capable of responding to various events in the kernel and allows user functions to be integrated without burdening the operating system. Here is a typical process for using eBPF:

1. Develop a script that interacts with data obtained as a result of events related to system calls.

2. Compile this script using standard compilers such as GCC or LLVM.

3. Load the compiled program into the system kernel after it has been successfully verified by the eBPF verifier.

4. Binding the loaded BPF programme to the required network interface, file, or process for its execution and processing in this context.

The **Fig 2.1** shows a program running in user space that includes an eBPF program for visibility at the process level in the Linux kernel.

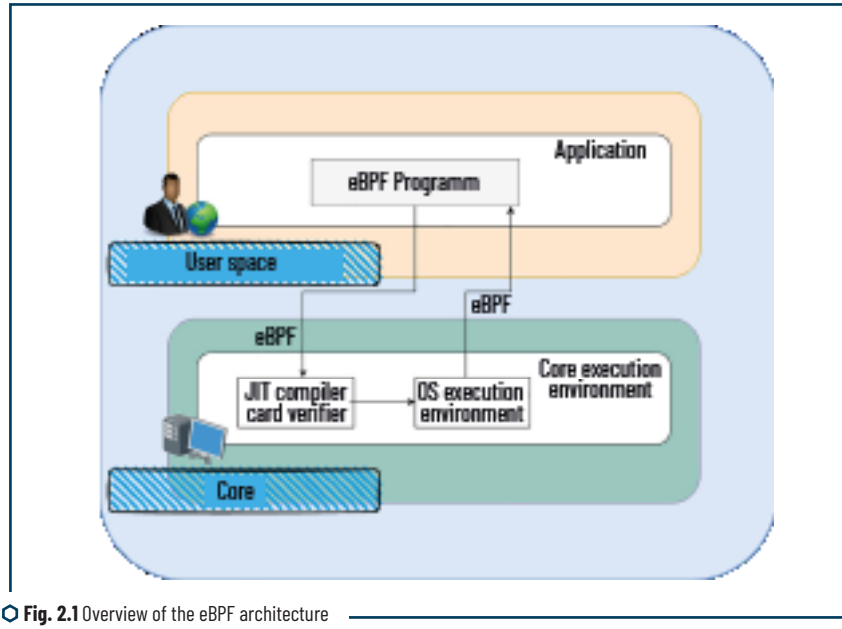

Fig. 2.1 Overview of the eBPF architecture

Typically, eBPF programmes written in bytecode are developed using higher-level programming languages such as Python or Golang, using a compiler compatible with eBPF bytecode. After that, the eBPF program is installed in the Linux kernel, where it is checked using the eBPF verification mechanism, which helps prevent potential errors. The program is then compiled and integrated with a specific event in the kernel, usually related to monitoring system calls. Each time such an event occurs, the program is activated, performs monitoring and analysis, and then sends the collected data back to the user application in user space. So-called "eBPF maps" [54] are used to communicate between the eBPF program and the user application.

eBPF Maps. Once the packet has been effectively processed and all the necessary data has been collected in the kernel space, the next step is to transfer this information to the user space. This is where eBPF maps come into play (**Fig 2.2**).

eBPF maps are types of data structures that can be used on both sides: from user space and within BPF programmes in kernel space. They function on a key-value principle and can be organised as hash tables, arrays or stacks. The size of an eBPF map must be predefined during the development of a BPF program

according to its needs, but it cannot exceed the 4 KB limit. Most map types support basic operations such as search, update, and delete, and another operation is available from user space: determining the next key. In general, the capabilities of eBPF maps are sufficient for most standard tasks [55].

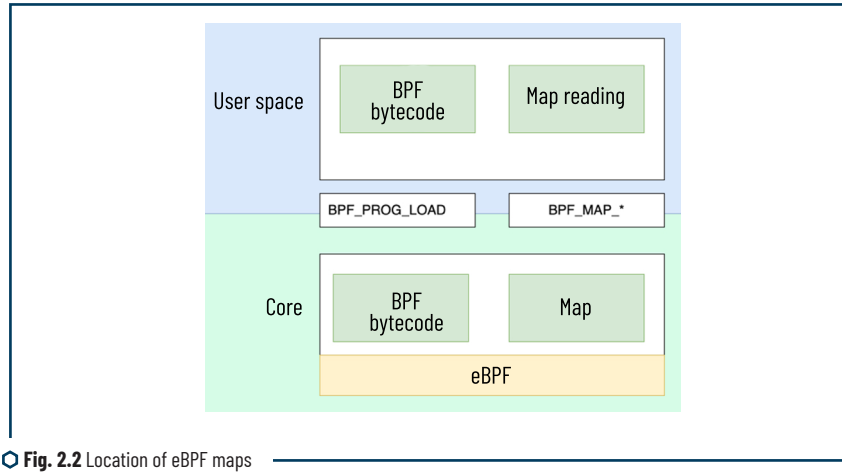

Fig. 2.2 Location of eBPF maps

Monitoring with outdated kernel, disk, and network tools. Classic packet sniffing has its drawbacks, especially in cloud environments. First, the process is expensive. It requires pattern matching for each packet, which consumes computing resources and can be ineffective at detecting threats. Second, a significant portion of modern traffic is encrypted. Let's Encrypt, and other services have made encryption more accessible and widespread, making packet analysis more difficult. The traditional method of packet capture is becoming less suitable for cloud and large-scale environments. Instead, eBPF provides a flexible, scalable, and efficient approach to network monitoring and security.

Disk forensics is used for incident investigations. During investigations, bit-for-bit copies of disks are collected to extract artefacts. However, artefacts can be random, and it is not always possible to determine exactly which data will be useful. Information stored in memory can be lost if it is not written to disk. Memory forensics focuses on new classes of attacks, but existing protection techniques complicate analysis. eBPF, in turn, provides the ability to monitor kernel events and log them with container attributes, thus creating a flexible and effective framework for container security.

Cloud approach. eBPF allows real-time monitoring of security events, transferring this data to the user space for analytical processing, incident investigation, or security strategy formation. Also, eBPF programmes support Kubernetes through monitoring programme APIs that use identification metadata to determine the relationship between events in the kernel and Kubernetes containers [56]. This enables detailed monitoring of the lifecycle of processes and network connections in cloud environments while minimising memory usage.

Virtual machine in the kernel. A pod is a set of Linux processes that run in the context of the kernel namespace. By executing system calls and queries to the operating system kernel where eBPF is located, the pod interacts with it. For example, the execve() system call is used to start a new process. When curl is launched from the bash shell, the bash process is copied, which calls execve() to launch the curl child process. With eBPF, you can intercept any kernel event, launch the corresponding program, obtain the required values, and pass them back to the user space. To illustrate, you can create an eBPF program that activates after the execve() system call completes; extracts metadata such as the name of the curl binary file, PID, UID, process arguments, and process capabilities in Linux; and then passes this data to user space, as shown in the **Fig. 2.3**.

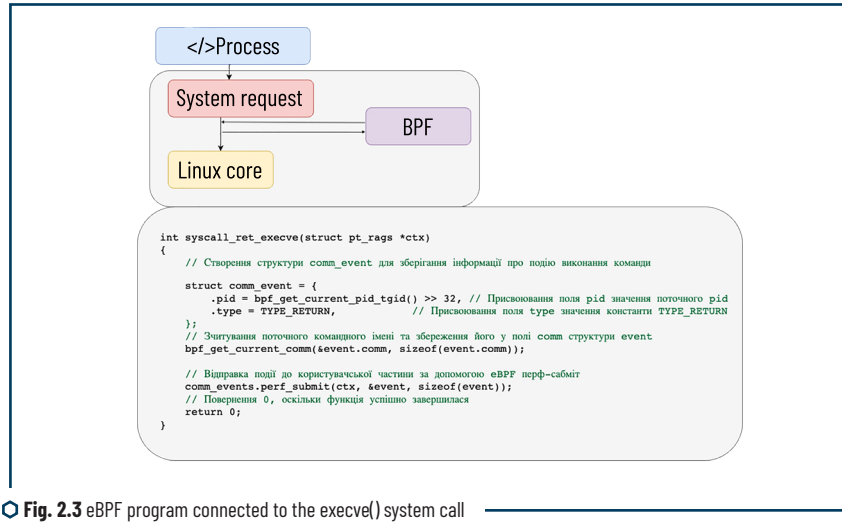

Fig. 2.3 eBPF program connected to the execve() system call

eBPF provides the ability to intercept any event at the kernel level and execute custom code using programmable logic. This can be thought of as a virtual machine functionality in the kernel with a universal set of 64-bit registers and eBPF programs tied to kernel code paths. eBPF programs can be dynamically loaded into and unloaded from the kernel. If such a program is bound to a specific event, it will be activated when that event occurs, regardless of what caused the event. eBPF offers a significant advantage over kernel updates or system reboots to access new features. This makes eBPF-based monitoring and security tools more effective and immediate, as they provide visibility into everything happening on the system. In containerised environments, this also includes visibility into processes in containers and on host machines (**Fig. 2.4**).

The internal activity of containerised environments helps detect ransomware without requiring modification of existing code or configuration. In addition, eBPF-based modules automatically monitor all applications and subsystems on the host machine from the moment they are loaded [56].

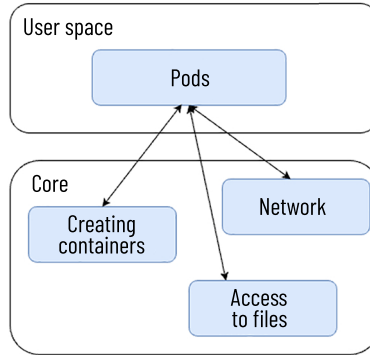

Fig. 2.4 An eBPF programme has kernel-level visibility into everything happening in all applications running on a Kubernetes node

2.2 METHODOLOGY FOR DEVELOPING EBPF–BASED MODULES

The BPF module development project consists of two types of source files. One is the source code file of the BPF program that runs in the kernel space. The other is the user program source code file, which is used to load the BPF program into the kernel, unload the BPF program from the kernel, and interact with the kernel state (for example, bpf_loader.c in the figure below) [56]. Currently, BPF programmes can only be developed in C, or more precisely in a limited C syntax, and the only one that can compile the source code perfectly is the clang compiler. clang is an external compiler for C, C++, Objective-C and other programming languages that uses LLVM as its internal interface (**Fig. 2.5**).

The target BPF file (bpf_program.o) is an ELF file, the contents of which can be read using the readelf command line tool. The llvm-objdump tool also provides information about the file:

llvm-objdump -d bpf_program.

The content output by llvm-objdump is actually BPF bytecode. The BPF program is loaded into the kernel as bytecode. When the BPF program is loaded into the kernel, the BPF virtual machine checks the BPF bytecode and runs JIT compilation to compile the bytecode into machine code. User space programs used to load and unload BPF programs can be developed in several languages, such as C, Python, Go, Rust, etc.

BPF programs. The Linux kernel is evolving rapidly, and the data types and structures in the kernel are constantly changing. Fields of the same structure type in different kernel versions may be reordered, renamed, or removed, replaced with completely different fields, and so on. For BPF programmes that do not need to look at the internal data structures of the kernel, porting problems may not arise. However, for those BPF programmes that need to rely on specific fields in the kernel data structure, it is important to consider

the problems that may arise in the BPF programme due to changes in the internal data structure of different kernel versions. To solve the problem of BPF portability in the kernel, two new technologies have been introduced: BTF (BPF Type Format) and CO-RE (Compile Once – Run Everywhere). BTF provides structural information to avoid dependence on Clang and kernel headers, while CO-RE makes compiled BPF bytecode portable, avoiding the need to recompile LLVM.

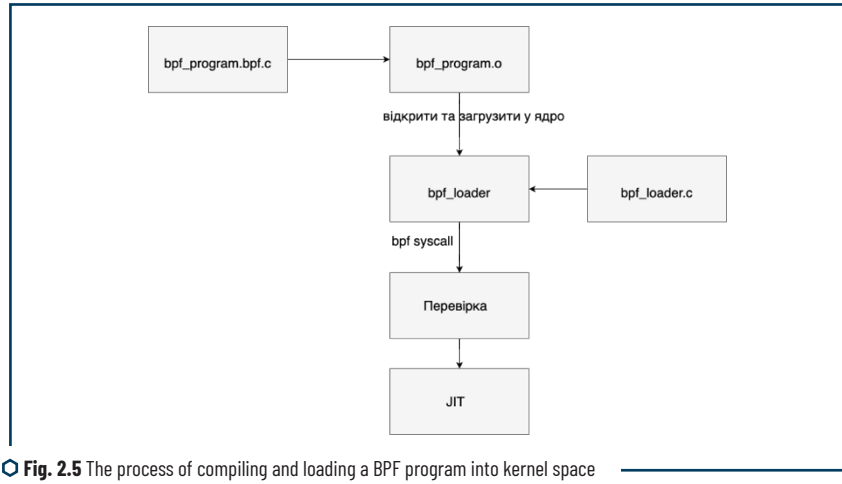

Fig. 2.5 The process of compiling and loading a BPF program into kernel space

BPF programs created using these new methods run on different versions of the Linux kernel without the need to recompile them for a specific kernel on the target machine. It is also not necessary to install hundreds of megabytes of LLVM, Clang, and kernel header dependencies on the target machine, as was previously the case.

Writing your first eBPF-based programme. Let's take a hello world BPF program and its user space loader as an example to see how a BPF program is implemented based on the structure proposed by libbpf-bootstrap. libbpf refers to tools/lib/bpf in the Linux kernel codebase, which is a C library. Bpftool is a bpf utility used in libbpf-bootstrap to generate xx.skel.h. base_program.bpf.c is the source code of the bpf program, which is compiled into the BPF bytecode ELF file base_program.bpf.o using the command clang -target=bpf. libbpf-bootstrap does not use the user space loader to directly load base_program.bpf.o, but instead generates the file base_program.skel.h based on base_program.bpf.o using the command bpftool gen. The generated base_program.skel.h file contains the BPF program bytecode and functions for loading and unloading the corresponding BPF program, which can be called directly from the user space program. In the BPF user space program, base_program.c, it is sufficient to include the base_program.skel.h file, load and connect the BPF program to the corresponding point in the kernel layer according to the set.

An example of developing an eBPF application based on libbpf-bootstrap. Note: Experimental environment – Ubuntu 20.04 (kernel version: 5.4.0-109-generic):

1. First of all, you need to install clang, a compiler for BPF programmes. It is recommended to install clang 10 and above, here is an example command:

apt-get install clang-10.

2. Download libbpf-bootstrap. This is a simple framework for development:

git clone https://github.com/libbpf/libbpf-bootstrap.git.

3. Initialise and update libbpf-bootstrap dependencies. libbpf-bootstrap has dependencies on libbpf and bpftool, which are configured in the project as a git submodule:

cat .gitmodules.

Therefore, before using the libbpf-bootstrap project to develop a BPF application, you need to initialise the git submodules and update them to the latest version:

git submodule update —init —recursive.

4. Using the libbpf-bootstrap framework, it is easy to add a new BPF application. To do this, go to the libbpf-bootstrap/examples/c directory, where you create two source files in C: base_program.bpf.c and base_program. (**Fig. 2.6**)

The logic of the bpf program in base_program.bpf.c is simple: insert bpf_prog into the execve call point (set by the SEC macro) so that every time execve is called, bpf_prog will be called back. The logic of bpf_prog is also very simple: it outputs a kernel debug log string. Compiling the new base_program is also not difficult, mainly because the libbpf_bootstrap project has a very extensive Makefile; we just need to add the base_program entry after the APP variable in the Makefile.

```
// libbpf_bootstrap/examples/c/Makefile
APPS = base_program minimal minimal_legacy bootstrap uprobe kprobe fentry.
```

After that, compile the base_program using the make command:

```
# Compiles the BPF programme BPF:
 @echo «Compiling BPF code...»
 @$(CC) -o .output/base_program.bpf.o src/base_program.bpf.c

# Generates a header file with a skeleton GEN-SKEL:
 @echo «Generating skeleton header file...»
```

```
 @bpf-gen-skel .output/base_program.bpf.o &gt; .output/base_program.skel.h

# Compiles the program object file CC:
 @echo «Compiling the programme...»
 @$(CC) -c src/base_program.c -o .output/base_program.o

# Creates an executable program file BINARY:
 @echo «Creating executable file...»
 @$(CC) .output/base_program.o -o base_program

# Default target for Make all: BPF GEN-SKEL CC BINARY
 @echo «Build complete.»
```

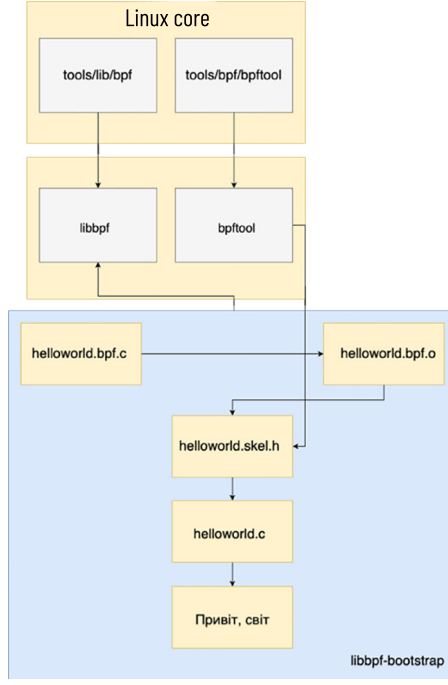

Fig. 2.6 EBPF program diagram using libbpf-bootstrap

The base_program programme must be run with root privileges:

 # Running the programme with administrative privileges
sudo ./base_program
 # Loading the BPF object of the programme from the buffer
libbpf: loading object 'base_program_bpf' from buffer
 # Finding the BPF programme in the tracepoint section for syscalls/sys_enter_execve
libbpf: sec 'tracepoint/syscalls/sys_enter_execve': found programme 'bpf_prog'
 # Processing relocations for the tracepoint section
libbpf: elf: section(9) '.reltracepoint/syscalls/sys_enter_execve', size 16
 # Displaying global data in memory
libbpf: map '.rodata.str1.1' (global data): at sec_idx 3, offset 0, flags 480.
 # Identification of map 0 as «.rodata.str1.1»
libbpf: map 0 is «.rodata.str1.1»
 # Displaying global data for base_program in memory
libbpf: map 'base_program.rodata' (global data): at sec_idx 4, offset 0, flags 480.
 # Identify map 1 as «base_program.rodata»
libbpf: map 1 is «base_program.rodata»
 # Collecting relocations for the system calls section
libbpf: sec '.reltracepoint/syscalls/sys_enter_execve': collecting relocation for section(2) 'tracepoint/syscalls/sys_enter_execve'
 # Relocation of pointer to data in BPF programme
libbpf: sec '.reltracepoint/syscalls/sys_enter_execve': relo #0: insn #9 against '.rodata'
 # Binding data to the BPF programme
libbpf: prog 'bpf_prog': found data map 1 (base_program.rodata, sec 4, off 0) for insn 9
 # Creating map '.rodata.str1.1'
libbpf: map '.rodata.str1.1': created successfully, fd=4
 # Creating map 'base_program.rodata'
libbpf: map 'base_program.rodata': created successfully, fd=5
 # Message about successful launch
Successfully started! Please run `sudo cat /sys/kernel/debug/tracing/trace_pipe` to see output of the BPF programmes.

The following command must be run in another terminal to view the output of the bpf program (when the execve system call occurs):

 # Execute the command to view the output of BPF programmes from the system trace buffer
sudo cat /sys/kernel/debug/tracing/trace_pipe
 # Event tracing output
 # The git process with PID 325411 on processor [002] invoked bpf_prog
git-325411 [002] .... 4769772.705141: 0: invoke bpf_prog: Hello, World!

 # Process sudo with PID 325745 on processor [005] invoked bpf_prog
sudo-325745 [005] .... 4772321.191798: 0: invoke bpf_prog: Hello, World!
 # Anonymous process with PID 325746 on processor [000] invoked bpf_prog
<...>-325746 [000] .... 4772322.798046: 0: invoke bpf_prog: Hello, World!

2.3 USING EBPF TO TRACK SYSTEM CALLS

Filtering system calls is an important means of protecting the operating system from potentially dangerous user programs. However, traditional approaches to this mechanism have their drawbacks. In modern container technologies, Linux system call filtering is widely used, which, unfortunately, does not always comply with security policy standards. The implementation of eBPF allows for the creation of more complex and adaptive security policies for Seccomp, taking into account dynamic conditions and variables. The combination of Seccomp and eBPF can be an effective tool for enhancing kernel-level security in systems.

Research shows that using eBPF for filtering can significantly improve existing security policies, for example, by reducing potential risks at an early stage of execution by up to 55% in the case of time specialisation, reducing actual vulnerabilities and increasing the efficiency of filters [8].

Systems run many untrusted applications on a trusted operating system kernel, which interact with it through system calls. Filtering these calls is a popular system call security mechanism that restricts their invocation according to predefined security policies. Early methods of filtering system calls use trusted user space agents to implement system call security policies. However, user space agents incur significant overhead for context switching to filter system calls. This is because each system call must switch to user space to check the policy, then switch back to user space to check the policy, and then switch back again.

Seccomp is a mechanism in the Linux kernel that provides the ability to restrict a program's access to system calls. Seccomp is an important tool for system call filtering because it allows you to specify which system calls are allowed to be executed on behalf of a particular program, thereby providing an additional level of security. For example, every Android application is restricted using Seccomp; systemd uses Seccomp to isolate user processes; Google's Sandboxed API project uses Seccomp to isolate C/C++ libraries; lightweight virtualisation technologies such as Google gVisor, Amazon Firecraker, Rkt, and Kubernetes use Seccomp. The main limitation of Seccomp is its lack of programmability for formulating advanced security policies. From its initial mode, which blocked all system calls except read(), write(), _exit(), and sigreturn(), Seccomp now (starting with Linux v3.5) allows writing custom security policies in classic BPF (cBPF) in filter mode.

Unfortunately, cBPF's programmability is too limited — security policies in Seccomp are mostly limited to static allow lists. This is primarily because cBPF does not provide a state storage mechanism, and therefore cBPF filters must be stateless. In addition, cBPF does not provide an interface for calling any other kernel utilities or other BPF programs. As a result, many desirable and/or necessary system call filtering features cannot be implemented directly on top of Seccomp-cBPF, but instead require significant kernel modifications (a major barrier to deployment). Recognising the need for more expressive policies, Seccomp

recently added a new feature known as Notifier to support the old idea of user space agents, which unfortunately shares their limitations.

It should be noted that opening the eBPF interface in Seccomp in early versions of Linux does not solve the problems due to:

1. The lack of basic features, such as serialisation.

2. The unsuitability of existing utilities, such as task storage, for Seccomp, as they are only aimed at privileged processes.

3. The danger of using Seccomp for existing functions, such as the current user memory access function, which does not solve the TOCTTOU problem.

To this end, a new type of Seccomp-eBPF programme was created, which can be easily programmed so that users can create advanced security policies for system calls in eBPF filter programmes. In particular, there is functionality for exposing, modifying, and creating new eBPF helper functions for secure management of filter state, kernel access, and user state, as well as kernel and user state synchronisation and the use of synchronisation primitives. Importantly, the system integrates with existing kernel privilege and capability mechanisms, allowing unprivileged users to safely install advanced filters and prevent privilege escalation. Seccomp-eBPF security is equivalent to two existing kernel components: Seccomp and eBPF. The system call monitoring module implements many features necessary for real-world use cases, such as user space checkpoint/restore, sleeping filter, and deployment configuration, to make Seccomp-eBPF a privileged feature. In addition, the existing container runtime has also been modified to support system call filtering based on Seccomp-eBPF. Seccomp-eBPF is used to implement various new features, including limiting the number and speed of system calls, flow integrity protection, and serialisation.

2.3.1 COMBINING SECCOMP AND EBPF POLICIES INTO A SINGLE SECCOMP–EBPF MODULE

Currently, Seccomp relies on the classic BPF language, which allows users to express system call filters as programs. It has a very simple set of register-based instructions, which makes filter programs easy to verify. Due to limited programming capabilities, cBPF filters in Seccomp mostly implement a list of permissions: the filter allows a system call only if the specified system call identifier is present. Sometimes a cBPF filter additionally checks the arguments of primitive data types and prevents the system call if the argument check fails. cBPF filters do not have states — the result of a Seccomp-cBPF filter depends only on a specific system call identifier and argument values (in the list of allowed ones), since cBPF does not offer any tools for state control. As mentioned at the beginning of this section, the size of an eBPF filter is limited to 4096 instructions, so complex security policies should be implemented using a chain of multiple filters. All filters installed in the chain are executed for each system call, and the action with the highest priority is returned. However, this chain behaviour is associated with increased performance, mainly due to indirect transitions. Seccomp decided to convert cBPF filters to eBPF code internally to take advantage of the widely optimised eBPF toolchain, which produces code that runs 4 times faster on x86-64 compared to direct BPF usage.

2.3.2 SECCOMP NOTIFIER

The limited expressiveness of BPF makes it difficult to implement complex security policies. Therefore, Seccomp has recently added support for a user space agent (called Notifier [57]) in addition to cBPF filters. Similar to earlier system call interception frameworks [57–60], it delegates decisions to a trusted user agent. Specifically, when Seccomp intercepts a system call, it blocks the task that invoked it and redirects the system call context (e.g., the PID and argument values) to the agent. The main drawback of Seccomp Notifier is significant delays due to additional context switches that occur when switching to and from user space. In addition, to check the contents of system call arguments that are pointers to user space, Seccomp Notifier must use ptrace to access the memory of the traced process. In addition to the performance impact, such verification is subject to race conditions from the time of verification to the time of use (TOCTTOU), in which the thread in the monitored program can change the memory contents (and thus the argument values) after the verification is completed by Seccomp Notifier. Finally, the need to run a user space agent in a trusted domain complicates its use in some deployment environments, such as non-daemon containers [61–65]. For all these reasons, Seccomp Notifier is not suitable for complex system filtering policies.

When tracking system calls with eBPF, it is important to focus on key call attributes to ensure detailed analysis of activity. The fields, that should be logged, noticed in **Table 2.1**.

Table 2.1 Module model

Field name	Description
TIMESTAMP	The time when the system call was made
PID	Identifier of the process that made the call
TID	The identifier of the thread within the process
UID	Identifier of the user under which the process was started
SYSCALL	System call name
ARGUMENTS	System call arguments
RETURN_VALUE	Value returned by the system call
STATUS	Execution status (e.g., SUCCESS, FAILED, BLOCKED)
ERROR_MESSAGE	Error message, if any
SOURCE_FILE	File from which the call was made
LINE_NUMBER	The line number in the file where the call was made.

Output example:

TIMESTAMP PID TID UID SYSCALL ARGUMENTS RETURN_VALUE STATUS ERROR_MESSAGE SOURCE_FILE LINE_NUMBER COMMENT
2023-10-26 13:15:25 3562 3563 1001 open /etc/passwd, O_RDWR -1 FAILED Permission denied

app.c 123 Trying to access passwd file
2023-10-26 13:16:01 3588 3589 0 write fd=3, buf=0x7ff... 42 SUCCESS — daemon_process.c 45 Writing to log

The data collected during ransomware attacks will be used as an information product for machine learning models to detect ransomware programmes.

2.4 MONITORING FILES AND CRYPTOGRAPHIC ACTIVITY USING HONEYPOTS AND EBPF

The functionality of eBPF is not limited to tracking. It can be used to respond quickly to suspicious activity by blocking potentially malicious processes or attacks at an early stage. This provides an additional layer of protection for systems, reducing the risk of damage or loss of important data. eBPF allows you to create hooks (programs) that can be associated with various events in the system, including file operations. For example, you can monitor system calls that perform file operations such as open, read, write, and delete. This section will describe an eBPF program whose task is to monitor the file system and detect ransomware activity. We propose developing not just a file system monitoring module, but a system of file lures based on ebpf. The use of honeypots — lure systems designed to attract and detect attackers — can be a valuable tool for detecting and analysing ransomware attacks. The eBPF programme can intercept low-level file system events and trigger alerts or responses when suspicious activity is detected. By combining honeypots with file system monitoring, organisations can create multi-layered protection capable of detecting and responding to attacker attacks in real time (**Fig. 2.7**).

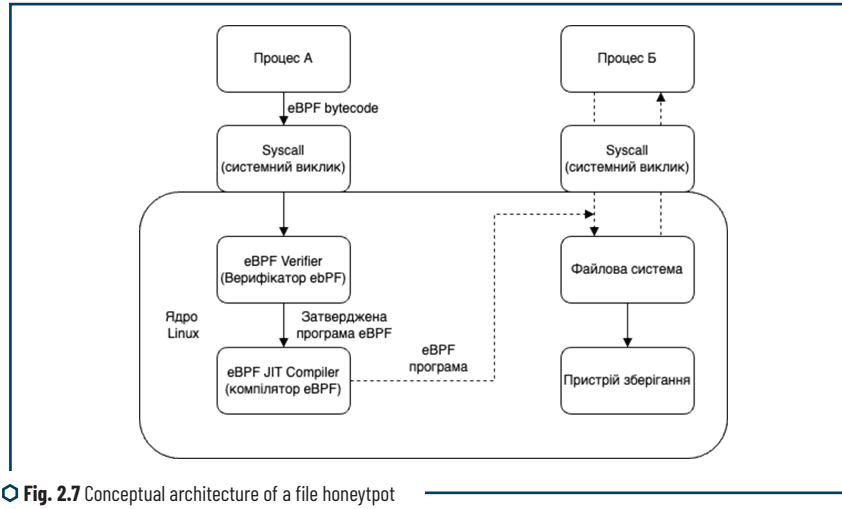

Fig. 2.7 Conceptual architecture of a file honeytpot

One of the key advantages of honeymots in combination with modern technologies such as eBPF is the difficulty of detecting them. Passive, indirect, and non-interactive detection methods allow cryptographers to avoid any network traffic or system calls that typically cause ransomware programmes to believe they are being analysed or detected. This feature makes it extremely unlikely that ransomware will be able to detect the presence of honeypots with high probability without leaving any traces of its activity. In addition, the proposed approach can be refined, optimised, and extended for specific environments and use cases. For example, the Python wrapper for eBPF can be modified to support additional features such as machine learning, statistical or behavioural algorithms for more advanced threat detection and correlation with other security information that can also be obtained using the eBPF subsystem (**Fig. 2.8**).

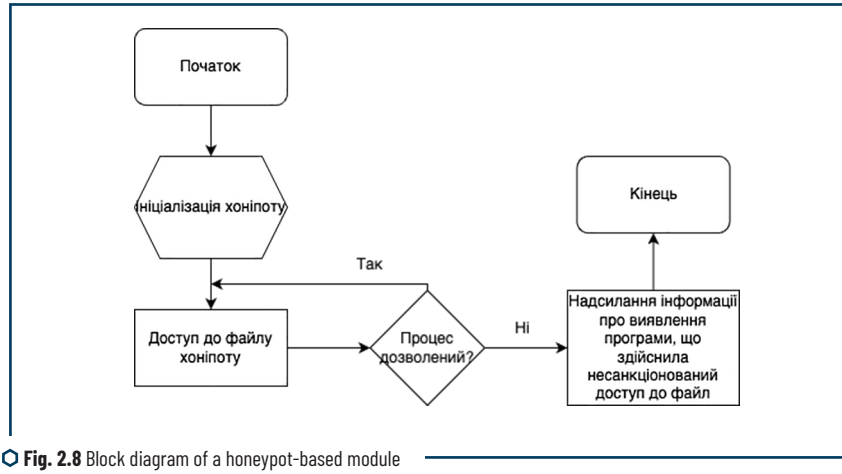

Fig. 2.8 Block diagram of a honeypot-based module

The selection of appropriate ransomware samples for testing ransomware detection systems based on honeypot-based methods is crucial for increasing the effectiveness and accuracy of the system's detection. To this end, the study identified a number of well-known and widely used ransomware families and selected samples from each family for testing. However, running these ransomware samples in a controlled testing environment was not always easy. Many modern ransomware strains have built-in "sandbox detection" features that allow them to detect that they are running in a virtual or emulated environment and modify their behaviour accordingly. This makes it difficult to accurately simulate a real-world attack scenario in a laboratory setting and to evaluate the effectiveness of honeypots in detecting and preventing such attacks. Despite these challenges, the study was able to successfully test the honeypot network against a number of ransomware samples and demonstrate its effectiveness in detecting and preventing such attacks. This highlights the importance of thoroughly testing and evaluating ransomware detection systems based on honeypot networks, as well as the need for ongoing research and development to stay ahead of the ever-changing threat landscape. Several recent ransomware samples were selected for testing: CryptoLock, AIRad, and DIMAQS.

BPF tracks calls to the open() system call, checks whether the name of the file being opened matches the name of the honeypot file, and if it does, sends an event to the policy engine. This means that the detection rate is not 0%, but may be higher if other legitimate processes or real users access this file. However, there is a list of allowed processes that can be defined to exclude certain processes that may trigger a security event, thus reducing the false positive rate. This configuration step requires a prior inventory performed on the host to identify and exclude such legitimate processes. Detecting the presence of honeypots on an endpoint can be a challenging task for ransomware [66]. eBPF-based "mousetraps" are also difficult to detect with high confidence, as the same set of tools used for our "mousetrap" purposes can be used for application/system monitoring, surveillance, and general troubleshooting by a system administrator (**Fig. 2.9**). Although it is possible to list all programs loaded into the kernel, it is highly unlikely that a program will be able to say with high confidence that a particular eBPF program is a "trap" because it looks like hundreds of other legitimate programs and tools that listen to syscall calls made by either the system, a program, or administrators [67–70].

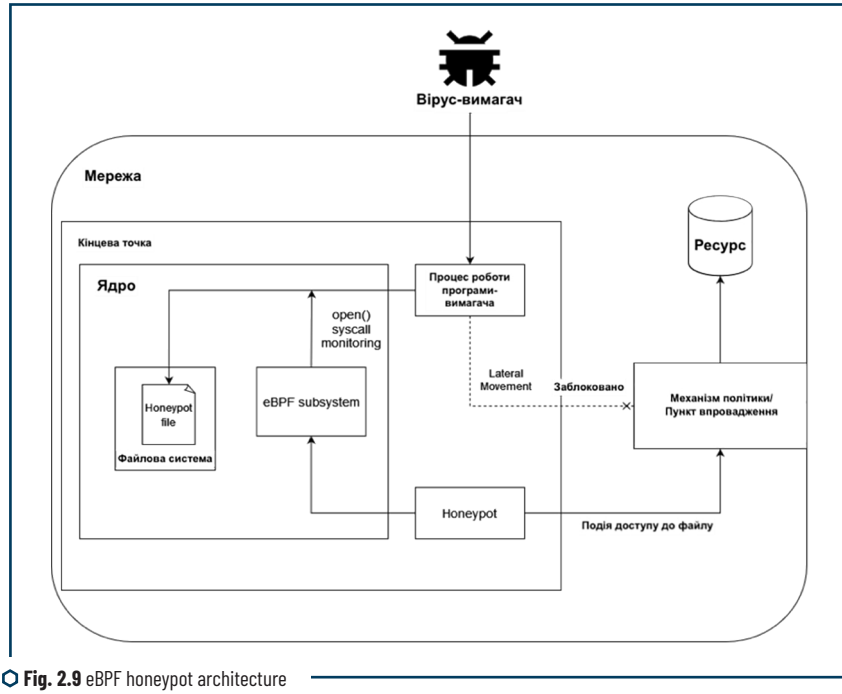

Fig. 2.9 eBPF honeypot architecture

For example, here is the result of running the bpftool command, which can be used to view a list of currently loaded eBPF programs (**Table 2.4**).

Table 2.4 Comparison of bpftool output for a legitimate system programme and a honeypot

Probe attached by the system	**Probe attached to honeypot with honey**

{
"id": 33,
"type": "cgroup_device",
"tag": "03b4eaae2f14641a",
"gpl_compatible": true,
"loaded_at": 1676542285,
"uid": 1000,
"bytes_xlated": 296,
"jited": true,
"bytes_jited": 166,
"bytes_memlock": 4096,
"map_ids": [1]
}
	{
	"id": 53,
	"type": "tracepoint",
	"name": "sys_enter_opena",
	"tag": "9081693d56ded011",
	"gpl_compatible": true,
	"loaded_at": 1676558452,
	"uid": 0,
	"bytes_xlated": 528,
	"jited": true,
	"bytes_jited": 315,
	"bytes_memlock": 4096,
	"map_ids": [6]
	}

There is a mention of the sys_enter_openat system call [71] (which is an interception point for the openat() system call, which in turn is called when the open() function is called by a user program), but it looks like any other programme that tracks files (for example, for logging purposes). Therefore, the mere presence of this information about the programme is not sufficient to determine whether it is a desirable target.

The study analysed the current state of ransomware attacks and their impact on organisations, and highlighted the limitations of traditional security measures in detecting and preventing these attacks. By studying honeyshops and their capabilities in detecting and preventing ransomware attacks, the study demonstrated the potential of file-based eBPF honeyshops as an effective tool in detecting these attacks, especially in the context of zero-trust architectures (**Table 2.5**).

Table 2.5 Event logging scheme

Field name	**Description**
TIMESTAMP	Time of the event. It is important to be able to track events in chronological order.
PID	Process identifier
PNAME	Process name. This can help quickly identify the application or service that caused the event.
UID	Identifier of the user who initiated the process
TYPE	The type of event (Open, Create, Delete, Encrypt, etc.)
FLAG	Event status (e.g. MIN — normal, MAJ — critical)
PATTERN	Sequence of actions that may indicate suspicious activity
TRESH_COUNT	Number of events over a certain period of time that may indicate abnormal activity
FILENAME	Path to a file or cryptographic function
RESULT	Result of the action (e.g. SUCCESS, FAIL, ERROR_CODE)

Output example:

TIMESTAMP PID PNAME UID TYPE FLAG PATTERN TRESH_COUNT FILENAME RESULT
2023-10-26 12:45:21 26858 MyApp 1001 Enc MIN — —-E EVP_EncryptInit_ex SUCCESS
2023-10-26 12:46:15 28178 FileApp 1002 Crea MIN OCD- 60 /tmp/tmp8imczwuu/Asia/Magadan.aes FAIL...

These fields provide additional information about events, promote a better understanding of the context, and help respond more effectively to suspicious or dangerous actions. This programme tracks cryptographic operations and file-related actions in real time, helping administrators and security analysts understand activity on the system.

2.5 NETWORK TRAFFIC ANALYSIS AND PROCESS MONITORING USING EBPF

XDP program type. One of the few types of BPF programs is XDP. It runs at an early stage of network packet processing. To mark a BPF program as XDP, the user must add the BPF_PROG_TYPE_XDP flag when loading into the kernel space.

The XDP program type allows you to perform various actions on a network packet. Here is a list of the most common ones:

— XDP_PASS — forward the network packet for further processing;
— XDP_DROP — drop the packet and terminate its execution at this point;
— XDP_ABORTED — stop processing the packet with the exclusion of the network packet trace point;
— XDP_TX — send back through the same network interface from which it came;
— XDP_REDIRECT — redirect the packet to another network interface.

XDP-type BPF programmes provide useful structures such as ethhdr for Ethernet packets and iphdr for IP packets for unpacking network packets. An XDP programme can be attached to the network interface to be monitored, or to several at once. This capability is key to detecting and countering viruses, as it provides the ability to respond to threats at the network level, rather than just at the operating system kernel level. This approach to responding to attacks allows them to be identified at the initiation or propagation stage, rather than being detected at the stage of file encryption or access violation.

You can disconnect a BPF program from the network interface using the bpftool tool. However, it is not possible to attach multiple XDP programs to a single network interface, as only one program can be attached to a single interface. Currently, XDP programs are the most common type of eBPF programs. It is also important to note the existence of special types of programs, such as "socket filter programs" and "socket option programs," which allow you to monitor activity in sockets and the network as a whole. They allow you not only to track transmitted packets, but also to change the parameters of specific sockets or connections. With such programmes, you can get a more detailed overview of network activity and respond to incidents by blocking traffic at the socket level.

There are already published studies demonstrating the use of eBPF/XDP in network monitoring and cybersecurity [72]. For example, there is a scenario for mitigating the effects of DDoS attacks, in which all malicious activity associated with DDoS must be blocked. In this context, the authors used eBPF/XDP to analyse incoming traffic and perform calculations in user space using heuristic algorithms, which, although not as accurate as full-fledged neural networks, do their job effectively. Based on the data obtained (e.g., identified malicious IP addresses), eBPF programs are created that refuse to accept traffic from these sources. With regard to observability in a microservices cloud environment, the ViperProbe framework [72] was proposed. This tool is designed to enhance both network and system-wide monitoring for by utilising various existing BPF types. In fact, eBPF can also be effectively used to track system events, such as system calls, in order to collect statistics and record all system behaviour. ViperProbe demonstrates limited resource consumption while providing effective analysis of eBPF metrics and Envoy metrics. The results showed that metrics collected using eBPF proved to be significantly more beneficial for implementing horizontal auto-scaling. In addition, it is worth mentioning the increasingly popular Cilium platform, which is an open-source application for network connectivity between applications and containers in which they are deployed using container management platforms such as Docker and Kubernetes. Cilium is based on a new Linux kernel technology called BPF, which allows powerful security control and management tools to be dynamically inserted into the Linux system itself. Since BPF runs inside the Linux kernel, Cilium security policies can be applied and updated without any changes to the application code or container configuration.

For example, consider a simple scenario: detecting viruses that interact with certain known command server IP addresses. To do this, you can create an XDP/eBPF program that will block all traffic to and from these IP addresses:

```
SEC("filter")
int block_ransomware_traffic(struct __sk_buff *skb) {
    bpf_hdr_pointer_t hdr_pointer = {};
    struct ethhdr *eth = bpf_hdr_pointer_advance(&amp;skb-&gt;data, &amp;skb-&gt;data_end, &amp;h-
dr_pointer, sizeof(*eth));
    if (!eth) {
        return XDP_DROP;      }
    // Check if this is an IP packet
    if (eth-&gt;h_proto != bpf_htons(ETH_P_IP)) {
        return XDP_PASS;      }
    struct iphdr *ip = bpf_hdr_pointer_advance(&amp;skb-&gt;data, &amp;skb-&gt;data_end, &amp;h-
dr_pointer, sizeof(*ip));
    if (!ip) {
        return XDP_DROP;      }
    // Check the sender's IP address
    if (ip-&gt;saddr == bpf_htonl(0xC0A80001)) {  // 192.168.0.1 as an example
```

```
        return XDP_DROP;   }
    return XDP_PASS;       }
```

This code uses eBPF to check each incoming packet and discards packets coming from the IP address 192.168.0.1.

For a network activity monitoring module, it is important to focus on aspects that reflect connection attempts, data transfer, and possible intrusions. Here are some fields that may be useful (**Table 2.6**).

Table 2.6 Network traffic logging scheme

Field name	Description
TIMESTAMP	Event time
SRC_IP	Sender's IP address
SRC_PORT	Sender port
DST_IP	Recipient's IP address
DST_PORT	Destination port
PROTOCOL	Protocol (e.g. TCP, UDP, ICMP)
PACKET_SIZE	Data packet size
FLAG	Packet flags (e.g. SYN, ACK, FIN)
STATUS	Packet status (e.g. SUCCESS, DROP, RETRY)
APP_LAYER_DATA	Application layer information (e.g., HTTP request type, DNS query)
ALERT	Signals about suspicious or abnormal activity

Output example:

TIMESTAMP SRC_IP SRC_PORT DST_IP DST_PORT PROTOCOL PACKET_SIZE FLAG STATUS APP_LAYER_DATA ALERT COMMENT
2023-10-26 12:45:21 192.168.1.5 54320 8.8.8.8 53 UDP 64 — SUCCESS DNS Query — Query to google DNS
2023-10-26 12:46:15 10.0.0.1 80 192.168.1.100 49582 TCP 512 SYN DROP HTTP GET HIGH Multiple SYN without ACK, possible SYN flood...

These fields provide a detailed overview of network activity, allowing specialists to quickly determine the type of activity and respond to potential threats.

Process execution monitoring is one of the security blocks during execution [122]. Process execution monitoring can be used to detect processes in a production environment that may be unexpected, or to detect execution patterns that should never occur. For example, a web server in a production environment

should never create a shell. Similarly, you may need information that the package manager is being called to install new dependencies on the host. Reviewing the arguments of the "curl" call can also help you understand what specific confidential data an attacker could have stolen. Another important part of process execution monitoring is the ability to enrich the context provided by notifications (regardless of their type). The user space process cache is used to provide the process tree. When analysing the process tree, the source of all processes affecting the initiated situations is revealed, regardless of the parent process. This is another example of the capabilities that arise from using security tools during execution: when entering the proc file system, you can observe that when a process terminates, its child processes suddenly switch to the process with ID [74]. This means that they recently restore the context of the parent process tree, although they could have been an important part of the context that could tell you which service or process was used. The process.ancestors selector in SecL syntax allows you to write a condition for one of the process parents, regardless of the number of intermediate processes that may have been added to try to bypass detection. To simplify the detection of web shells, you can use the following rule:

exec path of file in ["/bin/bash", "/bin/sh", ...] && ancestor process path of file == "/bin/my_web_server"

Another interesting advantage of diving deeper into the kernel than the system call level is the ability to work with pieces of information that are not normally even available in user space. For example, you can get a file layer in the overlayfs file system. This information has powerful security implications, as it can be used to determine whether the file to be executed was part of the base container image, or whether it has been modified (or simply created) compared to its original version in the base image. Detecting such a complex use case can be described with one simple rule:

exec . file . in_upper_layer == true

Similarly, you can also collect process credentials and supplement other events with them. This means that a complete set of user and group IDs, as well as kernel capabilities and executable file metadata, is collected. This allows you to write additional rules for processes with dangerously broad access, such as CAP_SYS_ADMIN, or simply detect the execution of binary files with setuid or setgid bit flags set.

For the "Process Monitoring Using eBPF" module, it is important to consider the basic attributes of processes to ensure a detailed and comprehensive analysis of their activity. Here are the fields that should be logged (**Table 2.7**).

These fields provide a comprehensive overview of process activity on the system and help identify anomalies or suspicious behaviour in processes. Example output:

TIMESTAMP PID PPID UID COMMAND_NAME CMDLINE STATUS CPU_USAGE MEMORY_USAGE START_TIME PRIORITY NETWORK_ACTIVITY FILE_OPERATIONS ERROR_MESSAGE COMMENT
2023-10-26 13:18:25 3652 1 1001 nginx nginx -g daemon off; RUNNING 5 % 80MB 2023-10-26 12:15

20 80Mbps /var/log/nginx/ — Web server process
2023-10-26 13:19:01 3678 3652 1001 nginx-worker — SLEEPING 2 % 20MB 2023-10-26 12:16 19 20Mbps /var/www/html/ — Nginx worker thread...

Table 2.7 Logging scheme for the process monitoring module

Field name	Description
TIMESTAMP	Event time
PID	Process identifier
PPID	Parent process identifier
UID	The user ID under which the process is running
COMMAND_NAME	Process command name
CMDLINE	Full command line of the process
STATUS	Process status (e.g., RUNNING, SLEEPING, ZOMBIE)
CPU_USAGE	CPU usage by the process
MEMORY_USAGE	Memory usage by the process
START_TIME	Process start time
PRIORITY	Process priority
NETWORK_ACTIVITY	Network activity associated with the process
FILE_OPERATIONS	File operations performed by the process

2.6 USING EBPF TO MONITOR PERFORMANCE METRICS

Performance metrics are key metrics that allow you to evaluate the performance of a computer system or program. They provide information about how the system uses its resources, such as processor time, memory, network bandwidth, and input/output (I/O) performance.

The impact of ransomware on performance metrics.

Ransomware is malicious software that encrypts user data and demands a ransom for its decryption. It can affect a number of performance indicators:

— CPU usage: ransomware viruses can intensively use the processor during file encryption, which leads to an increase in CPU load;

— I/O activity: encrypting and moving large numbers of files can lead to a significant increase in I/O activity, especially if the program works with large files;

— memory usage: some ransomware viruses can use a significant amount of RAM, which in turn can affect overall system performance.

Given this impact of ransomware on performance metrics, it may be important to detect them. In particular, monitoring abnormal changes in these metrics can serve as a warning of the potential presence of malicious software in the system. eBPF, thanks to its monitoring mechanisms, can be an effective tool for tracking such changes and detecting ransomware activity.

eBPF is a powerful tool for monitoring and analysing system events on the Linux kernel. With its help, developers can create programmes that interact directly with the kernel, allowing them to track various aspects of system activity, including performance metrics.

eBPF capabilities in performance monitoring:

1. Low overhead: compared to other monitoring tools, eBPF has relatively low overhead, making it effective for use in high-load systems.

2. Flexibility: eBPF programmes can be dynamically loaded and unloaded from the kernel, allowing monitoring to be configured according to user needs.

Examples of eBPF-based tools and frameworks:

— **BCC (BPF Compiler Collection)**: a set of utilities and libraries for developing, compiling, and running eBPF programs. In particular, it includes tools for performance monitoring, such as biotop (I/O monitoring) and cpudist (CPU usage distribution);

— **BPFtrace**: a high-level language for tracing based on eBPF, which allows you to quickly write scripts for analysing system behaviour.

In the context of ransomware monitoring, eBPF can be a key tool for detecting abnormal resource usage patterns, such as high CPU load or increased I/O activity, which may indicate malware activity.

For the "Using eBPF to monitor performance metrics" module, it is recommended to focus on key system performance metrics that affect overall performance and stability. eBPF allows you to obtain a detailed high-level data snapshot. Here are the fields that would be useful to consider (**Table 2.8**).

Table 2.8 Performance monitoring module logging scheme

Field name	Description
TIMESTAMP	Event time
CPU_USAGE	Total CPU usage in percent
MEMORY_USAGE	Total RAM usage in percent
DISK_IO	Disk input/output speed
NETWORK_IO	Network input/output speed
CACHE_HIT_RATE	Cache hit frequency
CONTEXT_SWITCHES	Number of context switches per core per second
LOAD_AVERAGE	Average system load for 1, 5, and 15 minutes
THREAD_COUNT	Total number of threads running
SYSTEM_CALLS_RATE	Number of system calls per second

Output example:

TIMESTAMP CPU_USAGE MEMORY_USAGE DISK_IO NETWORK_IO LATENCY ERROR_RATE CACHE_HIT_RATE CONTEXT_SWITCHES LOAD_AVERAGE THREAD_COUNT SYSTEM_CALLS_RATE COMMENT
2023-10-26 14:25:05 50 % 70 % 120MB/s 500Mbps 2ms 0.5 % 95 % 1000 0.70,1.20,1.50 300 2500 Normal performance
2023-10-26 14:26:10 80 % 90 % 150MB/s 1Gbps 5ms 1 % 90 % 2000 1.00,1.30,1.60 500 3000 High traffic detected

These metrics will help operators and engineers monitor system health, identify performance issues, and respond accordingly.

2.7 ACCESSING KERNEL DATA WITH EBPF

Today, BPF is a popular tool for solving various problems, including obtaining information from kernel data structures. There are various ways to use BPF to unload information from kernel data structures into user space.

Structure dumpers. Yunhong Song proposed virtual files that allow BPF programs to be integrated to create files in /proc for selected data. He created a new virtual file system for /sys/kernel/bpfdump, which reflects persistent data during mounting and allows the kernel to form subdirectories, for example, for BPF maps.

Among the patches is a new type of BPF programme, BPF_TRACE_DUMP, which outputs data via the bpf_seq_printf() and bpf_seq_write() functions. The programmes are loaded by default, with the option to pin them to /sys/kernel/bpfdump, such as "myps" in the /sys/kernel/bpfdump/task folder. Alan Maguire has developed a debugging method with printk() for detailed data output. Thanks to BTF (Binary Type Format), more information about structures can now be obtained in the kernel. Its changes allow structures to be printed with field names, which is useful not only for printk, but also for tools such as Ftrace. This demonstrates two ways to use BPF to access kernel data outside of the kernel.

BPF Iterator. There are several ways to dump kernel data into user space. The most popular of these is the /proc system. For example, 'cat /proc/net/tcp6' prints all tcp6 sockets in the system, and 'cat /proc/net/netlink' prints all netlink sockets in the system. However, the output format is usually fixed, and if users want more information about these sockets, they will have to make changes to the kernel, which often requires time for publication and release. The same applies to popular tools such as ss, where any additional information requires a kernel patch. To solve this problem, the drgn tool is often used, which extracts data from the kernel without modifying the kernel. But the main drawback of drgn is performance, as it cannot perform pointer tracing inside the kernel. In addition, drgn can produce incorrect results if a pointer becomes invalid inside the kernel. The BPF iterator solves the above problem by providing flexibility in what to collect with a one-time change for a given data structure in the kernel and performing all pointer tracing

inside the kernel. This flexibility is achieved by using bpf programs. Correctness is ensured by implementing pointer tracing inside the kernel with proper reference counting or lock protection. In its current state, the iterator only modifies a small portion of the data structures in the kernel.

The BPF iterator uses a kernel file to transfer data to user space. The data can be a formatted string or raw data. In the case of a formatted string, you need to use the bpftool iter subcommand to create a bpf iterator and bind it with bpf_link to a path in the BPF file system (bpffs). After that, you need to execute the 'cat <path>' command to print the results, similar to 'cat /proc/net/netlink'. For example, the following command is used to pin the bpf program in the bpf_iter_ipv6_route.o object file at the path /sys/fs/bpf/my_route:

$ bpftool iter pin ./bpf_iter_ipv6_route.o /sys/fs/bpf/my_route

Later, the results will be visible using the following command:

$ cat /sys/fs/bpf/my_route

To implement a bpf iterator in the kernel, the developer must fill in the following key data structure defined in the bpf.h file.

```
struct bpf_iter_reg {
        const char *target;
        bpf_iter_attach_target_t attach_target;
        bpf_iter_detach_target_t detach_target;
        bpf_iter_show_fdinfo_t show_fdinfo;
        bpf_iter_fill_link_info_t fill_link_info;
        bpf_iter_get_func_proto_t get_func_proto;
        u32 ctx_arg_info_size;
        u32 feature;
        struct bpf_ctx_arg_aux ctx_arg_info[BPF_ITER_CTX_ARG_MAX];
        const struct bpf_iter_seq_info *seq_info;      }
```

After filling in the data structure fields, you need to call 'bpf_iter_reg_target()' to register the iterator in the main bpf iterator subsystem. Below is a breakdown for each field in the bpf_iter_reg structure.

eBPF allows deep integration with the Linux kernel, providing the ability to monitor and analyse various aspects of system activity. Here are the fields that would be useful to log when accessing kernel data (**Table 2.9**).

Using eBPF to access kernel data allows for detailed analysis of system activity and the ability to take appropriate action on changes in the kernel.

Example output:

TIMESTAMP DATA_TYPE SOURCE_MODULE DATA_VALUE ACTION ASSOCIATED_PROCESS MEMORY_ADDRESS COMMENT
2023-10-26 14:30:05 Routing Table NET_CORE 192.168.1.1/24 via eth0 READ MyApp 0x3f78a2c5 Updated routing info
2023-10-26 14:31:10 Memory Stats MM_MODULE Total: 16GB, Free: 5GB, Used: 11GB READ SystemMonitor 0x4f88b2d7 Periodic check

Table 2.9 Log file scheme for the operating system kernel monitoring module

Field name	Description
TIMESTAMP	Event time
DATA_TYPE	Module or kernel component from which the data was obtained
DATA_VALUE	Value or content of the data obtained
ACTION	Action performed on the data (read, modify, delete, etc.)
ASSOCIATED_PROCESS	The process or application that initiated access
MEMORY_ADDRESS	The memory address where the data is stored (if known)

2.8 IMPLEMENTATION OF EBPF MODULES ON THE WINDOWS PLATFORM

The ebpf-for-windows project aims to allow developers to use familiar eBPF tools and application programming interfaces (APIs) on existing versions of Windows. Building on the work of others, this project takes several existing open source eBPF projects and adds plugins that allow them to run on Windows.

As shown in **Fig. 2.10**, existing eBPF tools (clang, etc.) can be used to generate eBPF bytecode from source code in various languages. The bytecode can be consumed by any program, either through bpftool or the Netsh command-line tool, which use a shared library that exposes the Libbpf API, although this work is still in progress.

The eBPF bytecode is sent to a static verifier (the PREVAIL verifier), which is located in a secure user-mode environment such as a system service (as is currently the case), an enclave, or a trusted virtual machine. If the eBPF program passes all verifier checks, it can be loaded into the kernel-mode execution context. This is typically done by JIT-compiling (using the uBPF JIT compiler) to native code, which is then passed to the execution context. In the debug build, the bytecode can be loaded directly into the interpreter (from uBPF in the kernel-mode execution context), although the interpreter is not present in the release build because it is considered less secure. See also the answer to the frequently asked question about HVCI below.

eBPF programs installed in the kernel-mode execution context can attach to various hooks and call various helper APIs available through the eBPF wrapper, which internally wraps the publicly available Windows

kernel APIs, allowing eBPF to be used on existing versions of Windows. Many helper functions already exist, and more hooks and helper functions will be added over time.

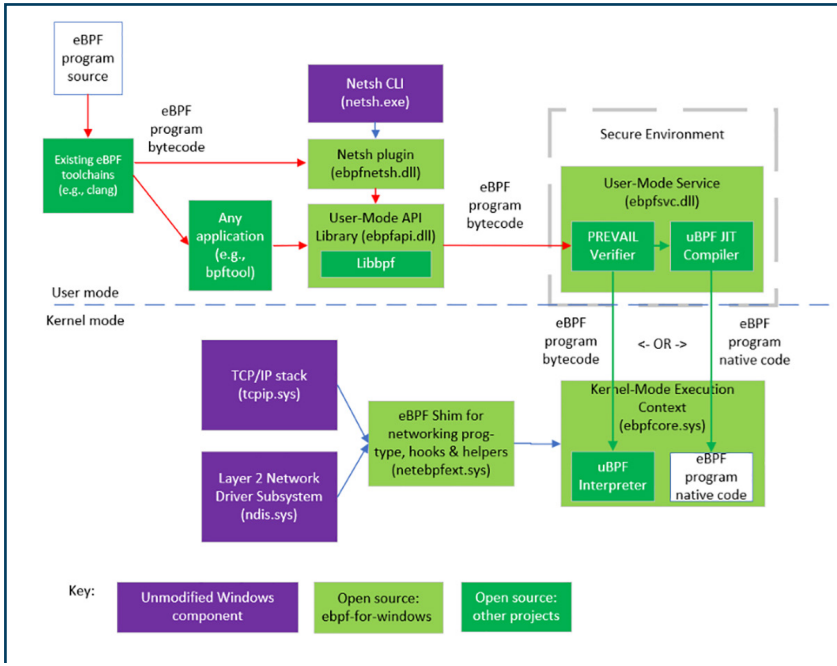

Fig. 2.10 Architecture of eBPF tools for Windows OS

3 CREATING AN INTEGRATED METHOD FOR ANALYSING RANSOMWARE BASED ON MACHINE LEARNING MODELS

3.1 ARCHITECTURE OF AN INTEGRATED RANSOMWARE DETECTION SYSTEM

The architecture of the integrated ransomware detection system presented in this paper is based on a flexible and extensible structure that uses various technologies to monitor and analyse cyber threats such as ransomware (**Fig. 3.1**).

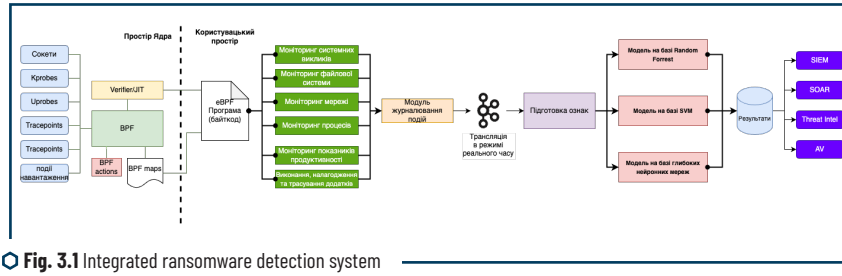

Fig. 3.1 Integrated ransomware detection system

The main components of the system include:

— monitoring of system events: BPF (Berkeley Packet Filter) is used, in particular BPF and BPF maps programmes, to collect data on system calls, network activity and processes. This allows the detection of abnormal behaviour that may indicate the presence of a ransomware virus [75];

— machine learning-based models: various models are used, including Random Forest, SVM (Support Vector Machine), and models for analysing neural networks, to analyse the collected data and classify behaviour as safe or potentially harmful;

— real-time data analysis and transport: the data transcription process is performed in real time, ensuring a quick response to potential threats;

— integration with other security systems: the system is integrated with higher levels of security, including SIEM (Security Information and Event Management), SOAR (Security Orchestration, Automation, and Response) platforms, Threat Intel, and antivirus software, enabling comprehensive protection.

3.2 DEVELOPMENT OF A MODEL PIPELINE

This section presents a strategic approach to machine learning that uses data from eBPF modules to effectively detect ransomware . The pipeline is designed to improve the accuracy of ransomware detection and strengthen system security. It includes five stages: event logging via eBPF, data normalisation, model

creation using SVM, model evaluation, and real-time threat detection [76]. Each step is important for optimising data and ensuring reliable detection.

Collecting and processing data from eBPF modules. Collecting data from different system levels and sources is a key step for effective monitoring and analysis. Using eBPF for this purpose allows you to obtain granular, high-quality data in real time. Here are the main steps of this process:

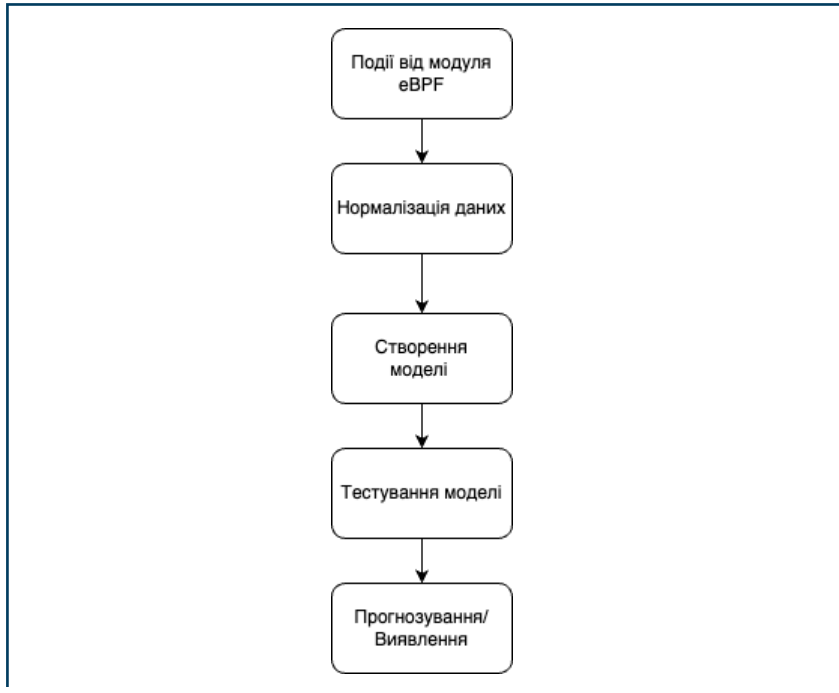

Fig. 3.2 Machine learning: model pipeline development

This diagram shows the standard process of data processing and machine learning model development:

1. Events from the module: this is the initial stage at which data or events from certain detectors or sensors are collected. For example, this could be information from surveillance cameras, motion sensors, etc.

2. Data normalisation (data preparation and feature engineering): at this stage, the collected data is processed and prepared for further analysis. The main goal is to transform "raw" information into useful characteristics (or "features") that can be used to train the model.

3. Model training and development: once the data has been prepared, it is time to develop and train the model. This involves selecting an appropriate algorithm, tuning parameters, and using the prepared data to "train" the model to recognise patterns or make predictions.

4. Model evaluation: after training the model, it must be tested on new, previously unknown data to assess how well it performs.

5. Prediction/detection: using the trained and tested model, predictions can be made on new data.

eBPF programmes monitor various system events and interactions, collecting output data that may signal potential malware activity [77]. This data includes system call patterns, file access paths, and other process metadata. The importance of these details collected through this process is key to analysing and identifying potential threats. The enriched and detailed data collected at this stage forms the basis for further steps, ensuring comprehensive analysis and accurate detection of ransomware. The next process is the normalisation of enriched data. This step is crucial for preparing data for machine learning models. It involves converting raw data into a consistent format and scale, making it more suitable for analysis [78].

Normalisation helps to eliminate any errors, biases, or anomalies caused by different scales and formats, ensuring that each feature contributes equally to the model's performance. This step improves the efficiency and accuracy of the machine learning model, paving the way for more reliable predictions and detections. Once the normalised data is ready, the next step is to feed this data into the machine learning model.

After training the model, it is important to test its performance. Model testing involves evaluating the model on a separate test dataset that it has not encountered before [79]. This step helps to assess the model's ability to generalise its knowledge to new, unknown data. Various metrics are used to measure model performance, such as accuracy, reliability, recall, and evaluation. The data obtained at this stage is used to improve and optimise the model for prediction and detection.

The final stage involves prediction and identification. Using an advanced model, eBPF data is processed in real time to predict and recognise potential cybercriminal actions. The model evaluates the data received, recognises patterns and behaviour, and predicts possible actions by malicious programmes. If malicious activity is detected, alarms are triggered and measures are taken to minimise risks. Such actions are critical to ensuring real-time protection of systems against ransomware, ensuring the security and stability of operations.

Identifying data sources.

First, you need to identify the data sources that will be monitored. These can be file systems (to track file access), network traffic, processes, or performance indicators.

Data sources used in this study:

— file system;
— file monitoring;
— monitoring of file and directory encryption activity;
— monitoring and analysis of network activity at the kernel level;
— process monitoring;
— monitoring performance indicators;
— monitoring access to the kernel.

As already described in the file system and cryptographic activity monitoring module, here are the fields collected by the module (**Table 3.1**).

Table 3.1 Module fields

No	Field name	Description	Example	Module name
1	TIMESTAMP	Event registration time	2023-10-26 14:30:05	All
2	EVENT_TYPE	Event type	Network Transmit	All
3	STATUS	Event status (successful, error)	SUCCESS	All
4	SOURCE_IP	Sender's IP address	192.168.1.10	Network
5	DESTINATION_IP	Recipient's IP address	192.168.1.55	Network
6	PROTOCOL	Protocol used (TCP, UDP)	TCP	Network
7	PACKET_SIZE	Data packet size	1500 bytes	Network
8	PID	Process identifier	1234	Syscalls
9	ACTION	Action (read, write)	READ	Syscalls
10	MEMORY_ADDRESS	Memory address	0x3F78A2C5	Syscalls
11	PROCESS_NAME	Process name	MyApp	Processes
12	USER_ID	Identifier of the user who started the process	1001	Processes
13	CPU_USAGE	CPU usage	35	Performance
14	MEMORY_USAGE	Memory usage	4GB/16GB	Performance
15	DATA_TYPE	Core data type	Routing Table	Kernel Data
16	SOURCE_MODULE	Kernel module or component	NET_CORE	Kernel Data
17	DATA_VALUE	Data value or content	Total: 16GB, Free: 5GB	Kernel Data

A specially developed Python script is used to prepare the data. Its main purpose is to convert the input log files into structured datasets that can be used for further analysis or model training.

Key constants:

— `TIME_PERIOD`: defines the time period for calculating the frequency of an event. The default value is one second;

— `TYPE_NAMES`: list of event types;

— `PID_OFFSET`: offset to avoid PID duplication during different detector runs;

— `LOGDIR` and `DATADIR`: paths to the log and data folders, respectively.

Functions:

— `counts(df)`: counts aggregated event statistics by various criteria (event type, time period, etc.). Forms an aggregated data frame based on the input data frame, grouping events by PID, event type, and time period;

— `sequences(df)`: analyses event sequences and counts their number. Forms a dataframe with event sequences (length 3) for each PID;

— `main()`: the main function that processes input log files and generates output datasets.

Basic logic:

1. The script processes log files from the "training" and "testing" folders.
2. For each log file, data is read and `PID_OFFSET` is applied to avoid PID duplication.
3. After that, the event type is replaced with the corresponding character code.
4. The `counts` and `sequences` functions are applied to calculate statistics and analyse sequences.
5. The results are saved to a CSV file.

Preparing data from log files. The script performs the following actions:

1. Sets constants to define the time period, event types, and other parameters.
2. Reads log files from the `training` and `testing` folders in the `../logs/` directory.
3. Changes the process ID (PID) for each log file to avoid duplication.
4. Converts event types from numbers to character labels.
5. Calculates the number of events grouped by type, period, and PID.
6. Counts sequences of events with a length of 3.
7. Combines the resulting data frames and saves the results in CSV format. The output data is stored in the `../data/` folder in two files: `training_data.csv` and `testing_data.csv`.

To use the script, you need to run it via the command line or another interface.

Data preparation process.

To ensure seamless analysis, it is important to refine and structure the raw data. A specialised Python script (dataprep.py) is used in the workflow to automate this process. Here is an overview of the data preparation stage based on sample output:

1. Script execution: the data preparation process began with the execution of the `./dataprep.py` command. This command activates the Python script responsible for pre-processing the data.

2. Feature columns: the data table consists of several columns, each representing unique features. These features include:

— `C_max`, `C_sum`: maximum and cumulative number of 'Create' actions;
— `D_max`, `D_sum`: maximum and cumulative number of 'Delete' actions;
— `E_max`, `E_sum`: maximum and cumulative number of 'E' ('Execute') actions;
— `O_max`, `O_sum`: maximum and cumulative number of 'Open' actions;
— `P_max`, `P_sum`: maximum and cumulative number of 'P' ('Process') actions.

We also have columns that represent action templates, such as:

— `CDO`, `COC`, `COO`, `DOC`, `DOO`, `EEE`, `OCD`, `OCO`, `OOC`, `OOO`: these columns denote sequences of operations. For example, `OCO` may represent the sequence of actions 'Opening', 'Creation', 'Opening'.

3. Indexing by process ID (PID): the data table uses the process ID (PID) as its index, making it easier to display and retrieve process-specific information. This facilitates the differentiation and analysis of each process's behaviour individually.

4. Interpretation: by observing the data, patterns associated with specific PIDs can be recognised. For example, PID `1019` shows a large number of `EEE` operations, indicating repeated actions. Some PIDs have a high frequency of certain patterns; for example, PID `30495` indicates a noticeable pattern of file operations with a large number of `COC`, `CDO`, and `OCO` sequences.

5. Post-processing analysis: once the data has been structured as shown in the table, it can be fed into further analytical tools or machine learning algorithms. By understanding the patterns in these features, especially file operation patterns, it becomes possible to detect anomalies or potentially malicious behaviour.

Ultimately, the `dataprep.py` script facilitates the organised presentation of complex system processes, making it possible to obtain information:

| Process ID | Max C | Sum C | Max D | Sum D | Max E | Sum E | Max O | Sum O | Max P | Sum P | KDO | KCK | CKO | DCK | DOO | EEE | OCD | OCO | OOOC | LLC |

|------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|

| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |

| 221 | 0 | 0 | 0 | 0 | 0 | 0 | 20 | 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 18 |

| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |

| 30495 | 233 | 1216 | 17 | 983 | 0 | 0 | 278 | 1458 | 17 | 983 | 983 | 228 | 5 | 949 | 34 | 0 | 983 | 233 | 39 | 201 |

| 30496 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

3.2.1 FORMATION OF DATASETS

Dataset and significance of features. Our experiments used a dataset of approximately 1.5 million events, containing 2,786 different processes, of which 53 were examples of ransomware. The dataset was divided according to Table 3.1. All ransomware samples used for the experiment were obtained from MalwareBazaar, a project whose main goal is to collect and share samples of malicious software. Successfully tested ransomware families include the following:

— IceFire;
— MONTI;
— REvil;
— AvosLocker;
— BlackMatter;
— HelloKitty.

Although many other families of ransomware viruses were considered, most of them either detected that the system on which they were running was a virtual machine, or were not compatible with our operating system, or did not meet the requirements for their launch (e.g., missing shared libraries), or lacked a live control server. The model assigns a certain weight to each feature based on its relevance in predicting the output. In our case, the most important features revolve around specific file operation patterns, namely the sequences of actions "Open", "Create" and "Delete".

Among these patterns, the three most influential features are:

1. Open, Create, Open (OCO): this pattern represents the sequence of opening a file, creating a new file, and then opening another file. It shows a specific behavioural pattern associated with ransomware activity.

2. Create, Open, Create (COC): this pattern involves creating a file, opening an existing file, and then creating another file. It captures a distinct sequence of file operations that ransomware samples often exhibit.

3. Create, Open, Open (COO): this pattern shows a sequence of creating a file, opening an existing file, and then opening another file. It represents another behavioural pattern that helps identify potential ransomware activity.

These patterns play a key role in distinguishing ransomware behaviour from normal system operations. Their high importance indicates that they provide strong discriminatory power for effective ransomware detection. Additionally, other important features include the number of deletion operations (D sum) and the maximum number of deletion operations (D max). These features capture the frequency and intensity of file deletion operations, which are often indicative of ransomware behaviour. When forming data sets, the main focus is on processing information from log files obtained from the `training` and `testing` folders. This information helps to create structured data frames that reflect statistical characteristics and event sequences.

Formation stages:

1. Log file reading: log files are read from the `training` and `testing` directories located in the `../logs/` folder.

2. PID correction: each log file may have its own unique PID, so to avoid duplication of PID values in the combined data frame, an offset `PID_OFFSET` is added to them.

3. Event type conversion: event types represented by numbers (0,1,2,3) are converted to character labels using the `TYPE_NAMES` list.

4. Event aggregation: using the `counts` function, events are grouped by PID, type, and time period. The result is an aggregated data frame that reflects the number of events for each type in each time interval.

5. Sequence formation: the `sequences` function creates a new data frame that considers sequences of events with a length of 3 events. This allows you to highlight the connections between sequential events.

6. Combining dataframes: the aggregated dataframe and the dataframe with sequences are combined to create the final dataframe, which contains all the necessary information.

7. Data storage: the final data frames are stored in the `../data/` folder in the `training_data.csv` and `testing_data.csv` files.

This process ensures that the datasets used to train and test the model are well-structured, consistent, and reflect real events and their sequences. To develop and test the effectiveness of a machine learning model that can distinguish ransomware from safe files, the key step is to correctly form the data sets. This ensures that the model will be trained and tested on relevant and representative data.

3.3 CHOOSING MACHINE LEARNING MODELS FOR DATA ANALYSIS

Depending on the nature of the data used to build the model (training data), machine learning algorithms are divided into four categories: supervised machine learning, unsupervised machine learning, semi-supervised machine learning, and reinforcement learning.

In supervised machine learning, the model is built by training on labelled data. In unsupervised learning, the model works with data points without any labels. The goal of an unsupervised algorithm is to organise data into clusters to describe its structure. This makes complex data simple and organised. Semi-supervised learning algorithms are between supervised and unsupervised learning algorithms. In this case, the algorithm is given some labelled data and works on unlabelled data. Reinforcement learning algorithms choose an action based on each data point and then evaluate how correct the decision was [80]. After that, the algorithm changes its strategy to learn better and achieve better results.

Machine learning with a teacher. Some examples of supervised learning algorithms are as follows:

1. Linear, logistic, polynomial regression, and least squares methods for regression problems.
2. *k*-nearest neighbours (*k*-NN) for classification.
3. Decision trees for classification.
4. Naive Bayes for classification.
5. Support vector machine (SVM) for classification.
6. Random forest for classification and regression.
7. Ensemble methods for classification and regression.

Unsupervised machine learning. In clustering, the goal is to identify inherent groupings in the data, and in association rule mining, the goal is to identify rules that describe large portions of the data [81]. Some examples of unsupervised learning algorithms are listed below:

1. *k*-means for clustering.
2. Hierarchical clustering for clustering.
3. Factor analysis for clustering.
4. Mixed models for clustering.
5. A priori algorithm for association tasks.
6. Principal component analysis (PCA).
7. Singular value decomposition.
8. Independent component analysis.

Most clustering algorithms are designed for numerical data that uses the concept of distance. Therefore, there are certain technical difficulties in applying clustering algorithms to cybersecurity problems with discrete data such as URLs, user names, IP addresses, port numbers, etc.

Reinforcement learning. This method, known as RL, effectively integrates the principles of dynamic programming and supervised learning, creating a powerful tool for analysing and solving problems [82]. It is a trial-and-error learning process for optimising the behaviour of an agent interacting with a stochastic environment. The agent performs action *A(t)* at each time step t, then perceives signal *S(t)* from the environment and receives reward *R(t)*.

The main components of a reinforcement learning system include the following elements:

1. Agent — an element that interacts with the environment and performs actions. An agent can be both a teacher and a decision maker.

2. The environment is the space in which the agent moves and learns. It responds to the agent's actions and presents it with new situations.

3. State (S_t) is the current situation that the environment returns to the agent. Each state is determined at a specific point in time t.

Reward (R_t) is the number that the agent receives as a result of performing an action. The reward can be positive or negative.

Action (A_t) is what the agent decides to do in a specific state. Performing an action affects the environment, which transitions the agent to a new state and provides a reward or penalty.

It is assumed that the agent begins interacting with the environment by receiving state S_t and selecting action A_t to be performed. After that, the environment returns a new state S_{t+1} and reward R_{t+1}, which help the agent learn to choose better actions in the future.

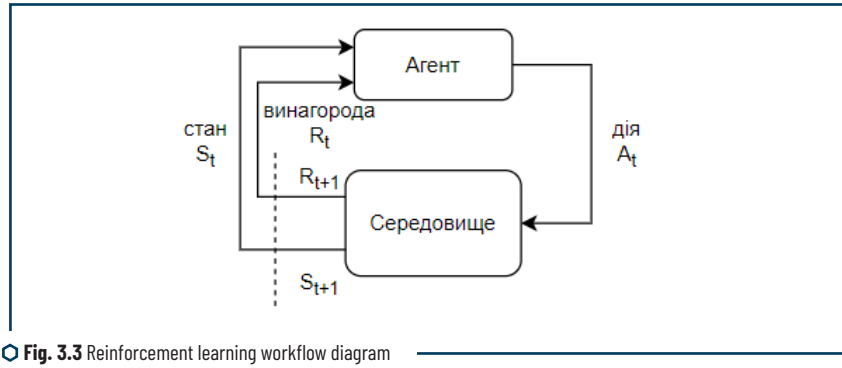

Fig. 3.3 Reinforcement learning workflow diagram

Reinforcement learning algorithms are used in various cybersecurity problems, such as network intrusion detection, anomaly detection, and others. Later, we will briefly discuss various machine learning models.

A naive Bayes classifier is used when there is a medium or large training data set with data points that have several attributes or features, and, given the classification parameter, the attributes are conditionally independent [82].

Some of the well-known applications of the naive Bayes classifier include:

1. Malware detection.
2. Spam detection.
3. Intrusion detection.
4. Detection of DoS and DDoS attacks.
5. Sentiment analysis.

Spam filtering is based on a naive Bayes classifier. This classifier divides emails into "Spam" and "Not spam". Email clients such as SpamBayes and SpamAssassin use this method [83]. In social networks, it analyses sentiment in status updates. Naive Bayes is well suited for discrete data, converges quickly, requires less data for training, and works well in multi-class tasks.

As part of efforts to effectively detect ransomware activity, a number of machine learning algorithms were analysed using data obtained from eBPF programmes. Each algorithm demonstrates specific characteristics for data-based analysis and prediction. When evaluating and testing different methods, the support vector machine (SVM) method proved to be particularly suitable for this task [84]. The choice of SVM in this research project is justified by its ability to work reliably and flexibly with data of varying complexity and dimensionality. The functionality of SVM to distinguish classes based on an optimal hyperplane that separates classes as efficiently as possible ensures its effectiveness in recognising complex patterns of ransomware behaviour. The main advantages of the method include its ability to effectively classify and perform regression analysis, as well as work with linear and nonlinear models through the application of various kernel functions. This approach contributes to high accuracy and reliability in detecting ransomware activity in real time, ensuring the protection and stability of computer systems. (**Table 3.2**).

Table 3.2 Evaluation of machine learning algorithms

Type	Name	Description
Classification	Random forest	Effectively processes large and complex data sets by modelling non-linear solutions
	Support vector machine (SVM)	Works well in high-dimensional spaces, ideal for binary classification
	Decision trees	Easy to understand and visualise, processes numerical and categorical data
Anomaly detection	Forest isolation	Effective for high-dimensional data sets. Specially designed for anomaly detection
	One-class SVM	Suitable for detecting outliers in high-dimensional data sets
	Local Outlier Factor (LOF)	Measures the deviation of the local density of a data point relative to its neighbours
Clustering algorithms	K-Means Clustering	Divides a dataset into K clusters. Can be used to detect unusual patterns
	DBSCAN	Does not require specifying the number of clusters. Can find clusters of arbitrary shape
Deep learning	Recurrent neural networks (RNN)	Ideal for analysing sequential data
	Automatic encoders	Used to detect anomalies through input reconstruction
Time series analysis	Long short-term memory (LSTM)	Effective for time series data

Supervised machine learning is the basis of the method of deciphering data about the relationship between input and output in various fields. It is based on the system learning from a dataset consisting of

paired input-output examples [85]. These datasets, characterised by labelled outputs, guide the learning algorithm to understand and learn complex relationships between input data and corresponding outputs.

When the output data is classified using discrete values representing different classes, supervised learning maneuvers towards classification tasks. Conversely, the presence of continuous output values directs learning towards regression tasks. The internal representation of input-output relationships in a learning model is represented by certain parameters. These parameters, which are crucial to the model's performance, are calculated during the learning phase, especially when there is no direct access to them.

The landscape of supervised learning is rich in a variety of algorithms, each with its own unique advantages. Among them, k-nearest neighbours (kNN) and support vector machines (SVM) occupy an important place. The kNN algorithm is based on the principles of proximity. This method determines the categories of new data depending on their spatial proximity to already known labelled data, using the most frequent class labels among the k nearest neighbours. This algorithm does not use parameterisation, but instead classifies based on the distances between the new data and the existing examples in the training set.

3.3.1 DEVELOPMENT OF A MODEL FOR CLASSIFYING RANSOMWARE USING DECISION TREES AND RANDOM FOREST ENSEMBLES

Decision trees are a model that responds to sequential questions and provides a specific path that will lead to results. The model has several "if this, then that" conditions, which ultimately allow you to get the expected result [86].

Advantages of using decision trees:

1. Easy to interpret and create simple visualisations.

2. Internal processes can be observed.

3. They work well with large data sets and are extremely fast.

Disadvantages of decision trees:

1. Decision trees are prone to overfitting, especially when they have many layers and complex structures. Overfitting occurs when a decision tree becomes too specific to the data set on which it was trained and loses its ability to generalise. This can lead to poor predictions or decisions when the tree is used on new, unknown data.

2. Decision trees may have difficulty modelling certain types of relationships between data, especially when these relationships are complex or non-linear. They work well for simple, hierarchical decision structures, but may not effectively handle tasks where the relationships between variables are more complex or where a large number of interrelated factors need to be taken into account.

The decision tree for detecting ransomware, created using eBPF modules, consists of the following steps:

1. File system monitoring: At this stage, the system uses the eBPF module to monitor activity in the file system. This includes checking for unusual file changes, write speeds, and other suspicious patterns that may indicate the presence of a ransomware virus.

2. Analysis of unusual activity: if unusual activity is detected, the system proceeds to detect the ransomware. If no unusual activity is detected, the process continues with performance monitoring.

3. Performance monitoring: at this stage, the system checks for unusual consumption of system resources such as CPU, memory, or network activity, which could be a sign of malicious activity.

4. Resource consumption analysis: if unusual resource consumption is detected, the system proceeds to ransomware detection. If not, it proceeds to system process monitoring.

5. System process monitoring: at this stage, the system analyses the activity of system processes, looking for suspicious or unusual processes that may indicate the presence of a virus.

6. System process analysis: if suspicious processes are detected, the system proceeds to ransomware detection. If no suspicious processes are detected, the system is considered safe.

7. Ransomware detection: at this stage, the system identifies potential ransomware based on the collected data and proceeds to apply security measures.

8. Application of security measures: after identifying the ransomware virus, the system takes the necessary measures, such as isolating the infected process, notifying the system administrator, and initiating recovery processes.

This decision tree allows the system to effectively detect and respond to ransomware using various monitoring and analysis modules, providing a multi-layered approach to cybersecurity.

Random Forest mathematical model.

The decision tree works by recursively dividing the data space based on features. Metrics such as the Gini index (G) or information gain are used to determine the best division:

1. The Gini index for partition S and feature X is calculated using the formula:

$$G(S,X)=\sum_{i=1}^{c}\left(p_i\right)^2, \tag{3.1}$$

where is the proportion of instances of class i in subset S.

2. The information gain is defined as the difference between the entropy before the split and the weighted entropy after the split:

$$Gain(S,X)=Entropy(S)-E\_(v\in Values(X))(S/(S\_v)\times Entropy(Sv)), \tag{3.2}$$

where is a subset of data where feature *X* takes the value

Random forest.

A random forest creates an ensemble of *B* decision trees, each of which is trained on a subset of data obtained by bootstrapping, and each split in the tree is made on a subset of features. For a classification task, the random forest prediction is equal to the mode of the tree predictions:

$$RF(x)=mode\{DT1(x),DT2(x),\ldots,DTB(x)\} \tag{3.3}$$

For the regression task, the random forest returns the average value of the tree predictions:

$RF(x)=B/1 \sum_{b=1}^{B} DT_b(x)$, (3.4)

Here, $DT_b(x)$ is the prediction of tree b on input data x.

These formulas give a general idea of how a random forest generates its predictions. In addition, there are many additional parameters and optimisations that can be used when creating a random forest.

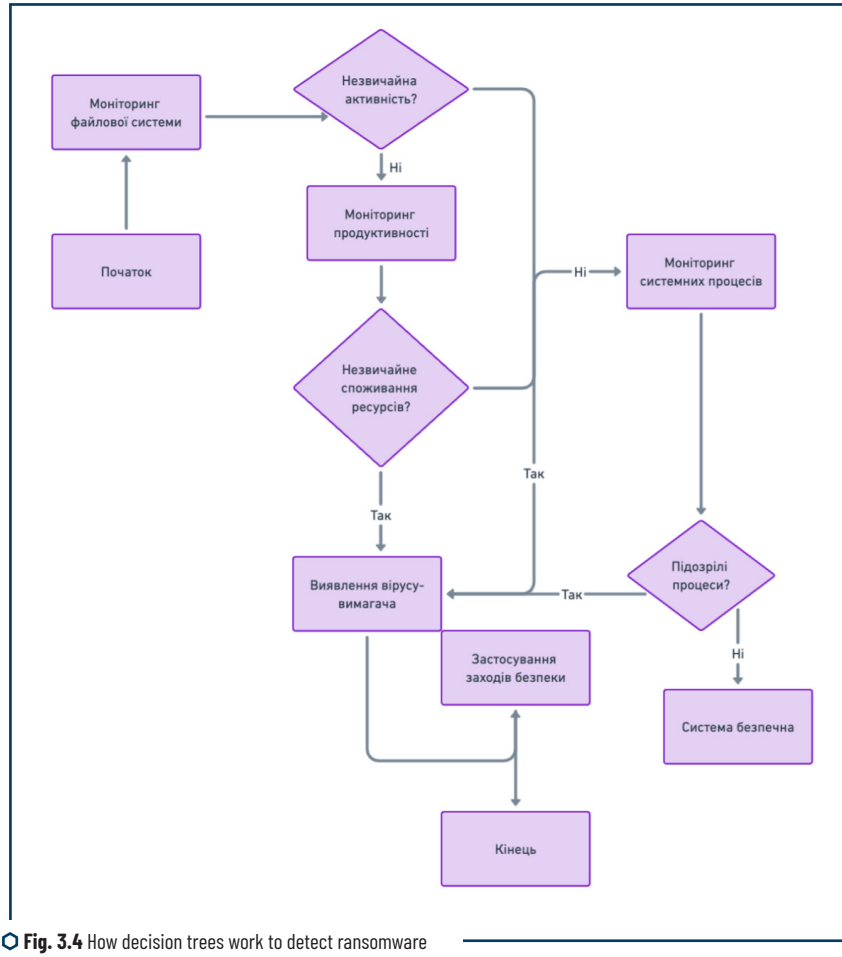

Fig. 3.4 How decision trees work to detect ransomware

Let's look at some additional parameters and optimisations that are commonly used when creating a random forest:

1. n_estimators. Specifies the number of trees in the forest. Increasing this parameter can improve the model's performance, but it also increases the training and usage time of the model.

2. max_features. The number of features considered when searching for the best split. Decreasing this number can make the algorithm more stable at the cost of a slight decrease in performance.

3. max_depth. The maximum depth of the tree. Limiting the depth can help avoid overfitting.

4. min_samples_split. The minimum number of samples required to split an internal node. Helps control the detail of the tree.

5. min_samples_leaf. The minimum number of samples required for a leaf node. Setting this parameter can help reduce the number of small branches that can cause overfitting.

6. bootstrap. Use bootstrapping to create trees. If the parameter is set to False, the entire dataset is used to construct each tree.

7. oob_score. Whether to use out-of-bag samples for performance evaluation. Out-of-bag samples are samples that were not used during the construction of a particular tree.

8. class_weight. Class weights for classification tasks if classes are unbalanced.

9. n_jobs. Number of cores for parallel training. Typically used to speed up training on multi-core machines.

These parameters can be optimised using methods such as cross-validation and grid search to obtain the best possible model performance on the data.

Training and evaluation. When training a random forest, it is important to consider a number of parameters, such as tree depth, number of trees in the forest, and maximum number of features to consider. To evaluate the quality of the model, you can use cross-validation and various quality metrics, such as accuracy, completeness, and F-measure.

Feature importance. Random forests allow you to evaluate the importance of each feature. This is determined based on how often the feature is used for splitting and how much it improves the quality of the split.

Thus, decision trees and random forests are powerful tools for solving machine learning problems. They provide an intuitive understanding of the data, the ability to evaluate the importance of features, and flexibility in modelling complex relationships between features.

3.3.2 THE SUPPORT VECTOR METHOD AS A TOOL FOR SEPARATING "SAFE" AND "UNSAFE" PROGRAMMES BASED ON A HYPERPLANE THAT MAXIMISES THE DISTANCE BETWEEN CLASSES

The Support Vector Machine (SVM) method is the most attractive machine learning method. SVM is an effective classification method that finds application in various fields. SVM can be useful in cybersecurity if applied to improve the accuracy of intrusion detection systems. This classifier helps in detecting malicious network traffic [87, 88]. Although binary SVMs typically have higher accuracy, the creation of effective one-class SVMs could reduce the need for the labour-intensive process of manually labelling datasets. Single-class SVMs only require a training dataset containing normal traffic.

Feature space is a set or collection of features used to categorise data. A hyperplane is a straight line, plane, or any other non-linear boundary drawn in feature space in such a way that the entire set of malicious programs remains well separated from the safe ones. Hyperplanes are easy to construct when data sets are sparse. In cases where data sets are dense, it can be difficult to separate samples that behave differently from each other. In such cases, it is necessary to construct a nonlinear hyperplane. Kernel functions in SVM are used to solve nonlinear problems by converting them into a linear problem, which is done by mapping the low-dimensional input space to a higher-dimensional space [89].

To detect malicious software, the first step is to effectively extract features. The training data set is prepared by loading samples from any data repository, and then the relevant features are extracted from the samples. SVM, as a supervised machine learning method, requires that the data be labelled as malicious or benign software in order to classify it during the training phase. These features, together with the labels, are used to construct a data sample in feature space to find the optimal hyperplane that effectively separates malicious software from benign software. Studying the hyperplane is the most important phase of SVM. Optimisation techniques are used to reduce the number of non-zero weights to a few that correspond to important characteristics that determine how the hyperplane should be constructed. Linear algebraic methods are used to study the hyperplane. Kernels are functions that determine the similarity between two input data. In cases where feature vectors are very dense and it is difficult to find a linear hyperplane, SVM uses a kernel method to map feature vectors into a higher-dimensional space.

The idea of using kernel functions is based on the fact that many algorithms, such as SVM, can be formulated in such a way that the only way they interact with data is by calculating the dot products of the feature vector of data points. Similarity functions are proposed to overcome some of the limitations of kernel functions. In this case, similarity features are calculated using a similarity function and added together. The similarity function measures how similar each instance is to a given landmark. Some applications of SVM in cybersecurity are intrusion detection systems, cyberattack classification, and malware detection. For IDS, you can have single-class, two-class, or multi-class SVM models [89]. A single-class SVM requires only a training data set containing normal traffic, while a two-class SVM requires a training data set containing both normal and abnormal traffic. Multi-class models can be used to classify malicious traffic into one of several classes, such as denial-of-service attacks, distributed denial-of-service attacks, probing attacks, malware attacks, etc. [90].

Feature selection is an important step when it comes to detecting malware using machine learning. Some of its goals are to reduce noise, improve the accuracy and speed of classifier training, and minimise the data set for selecting the best training set. The methods are based on minimising generalisation boundaries using gradient descent and are computationally feasible. SVMs may perform poorly in situations with a large number of irrelevant features. Typical raw sample data includes information about headers, possible packers and compressors, the size of different sections, lines, entropy, file hash, file size in bytes, etc. The goal is to convert raw characteristics into numerical features. Extracting numerical features from the obtained data is an important part of the machine learning process. Some other features are the size of the code in the file, the number of resource languages detected in the file, the number of resource types detected in the file, the decimal value of the entry point, i.e. the place in the code where control is

transferred from the host OS to the file, the size of the initialised data, the length of the "original file name, the size of the part of the file that contains all global, uninitialised variables or variables initialised to zero, and so on. Some of the characteristics that can be extracted from malware are API calls, permissions, opcodes, system calls, CPU and RAM usage, memory usage, network usage, and so on. To perform data sampling and/or classification, it is necessary to create subsets of the complete data set.

The SVM classifier learns by determining the optimal hyperplane for the best separation of data. Using a kernel function, the data is projected onto a higher-dimensional feature space. The main goal is to find a hyperplane that classifies the training data set into two classes: benign and malicious, based on the labels provided. SVM, being a supervised machine learning technique, uses data that is pre-labelled during the training phase. Feature selection methods aim to reduce the dimensionality and increase the compactness of the feature vector representing the files.

Supervised learning has a variety of algorithms, each with unique advantages. Among them, k-nearest neighbours (kNN) algorithms and support vector machines (SVM) occupy a special place. The kNN algorithm determines the categories of new data points based on their spatial proximity to already labelled examples, assigning a classification according to the dominant class among the k nearest neighbours. This method is non-parametric and is based on measuring the distances between a new point and all labelled training examples. SVM, in turn, is renowned for its ability to reliably classify and perform regression analysis by determining the optimal hyperplane for clear separation of data classes with the largest possible gap. It achieves this by transforming the data into a measurable feature space and carefully constructing a decision boundary that maximises the difference between individual classes. Its flexibility in handling both linearly and non-linearly separated data is further enhanced by various kernel functions [91].

In the context of this research project, the main focus was on testing the performance of SVM using both a linear kernel and a radial basis function (RBF) kernel. The experiments and research in this project aim to shed light on various aspects of SVM's capabilities with these kernels, providing a comprehensive understanding of its functioning and effectiveness.

To detect ransomware using the support vector machine (SVM) method, both linear and non-linear SVMs can be used, depending on the complexity of the data. SVM is effective in classification when class separation requires finding the optimal separating hyperplane. Here is a proposed general description of the SVM mathematical model for two-class classification of ransomware:

1. Problem formulation.

SVM searches for a hyperplane that optimally separates the two classes of input data, minimising classification error.

Classes here can be defined as $y_1 = 1$ for ransomware viruses and $y_1 = -1$ for safe programmes.

2. Margin maximisation formula.

The main goal of SVM is to maximise the margin between the closest data points of both classes (support vectors) and the hyperplane. This formula reflects the SVM objective function, which aims to minimise the square of the norm of the weight vector w. The smaller the norm w, the greater the margin between the support vectors and the separating hyperplane. This ensures better model generalisation on new data. The minimisation function looks like this:

$$min\_(w,b),1/2\|w\|2. \quad (3.5)$$

3. Introduction of relaxation variables.

In cases where linear separability is impossible, relaxation variables ξi are introduced to allow for classification errors:

$$min\_(w,b)1/2w2+C\sum i=1ni,$$

where *C* is a regularisation parameter that controls the trade-off between increasing the margin size and minimising classification errors.

4. Kernel trick.

To solve non-linearly separable problems, a kernel trick is used, which allows SVM to work effectively in higher-dimensional spaces. A kernel function, in particular a radial basis function (RBF), is used to transform the input data into a higher-dimensional space where it can be linearly separated. The parameter γ controls the width of the "Gaussian" kernel, affecting the smoothness of the decision boundary. The kernel function can be chosen, for example, as a radial basis function (RBF):

$$K(xi,xj)=exp(-xi-xj2), \quad (3.6)$$

where γ is a parameter that needs to be tuned.

5. Solving the optimisation problem.

The optimisation problem is usually solved using quadratic programming methods, and the solution gives the coefficients w and b for the hyperplane and the indicators of the support vectors.

6. Classification of new data

This formula determines how the classifier outputs the classification decision for a new input vector *x*. The sign of the linear combination of weights, input data, and bias indicates which class the vector belongs to. The classification of new input data x is performed using the sign of the function decision:

$$f(x)=sgn(w^T x+b). \quad (3.7)$$

This code is used to classify data using the support vector method and perform the corresponding analysis.

1. Importing modules:

— `argparse` is used to analyse command line arguments;
— `numpy` and `pandas` for data processing;
— `sklearn` for modelling and visualisation;
— `matplotlib` for creating graphs;
— `joblib` for saving and loading models.

2. Data structure:

— defined paths to files with data, labels, and models.

3. Functions:

— `get_labels()`: obtains class labels;

— `best_features_linear()` and `best_features_rbf()`: visualise feature importance for linear and RBF kernel SVMs, respectively;

— `refit_strategy()`: determines the best parameters for the model based on cross-validation results;

— `train()`: trains the SVM model on training data;

— `test()`: tests the model on test data and visualises the results using a confusion matrix and ROC curve.

4. Main code block `main()`:

— parsing command line arguments to determine the operating mode (training or testing) and the path to the label file;

— execution of the `train()` or `test()` functions depending on the selected mode.

5. Condition `if __name__ == "__main__":`:

— launches the main code block when the script is executed as the main programme.

In general, this code allows the user to train an SVM-based classification model for a given dataset, determine the importance of features, and evaluate its performance on a test dataset.

Data preparation:

```
# scale the training data
scaler = StandardScaler().fit(X_train)
scaler.transform(X_train)
```

Searching for hyperparameters for SVM:

```
kernel = ['rbf'] # 'rbf' or 'linear'
class_weight = [{1: w} for w in np.linspace(10, 200, 10)]
C = np.logspace(-1, 0, 10)
param_grid = dict(class_weight=class_weight, kernel=kernel, C=C)
scores = ['recall', 'balanced_accuracy']
grid = GridSearchCV(svm.SVC(max_iter=1_000_000), param_grid=param_grid, scoring=scores, refit=re-
fit_strategy)
grid.fit(X_train, y_train)
```

Feature importance evaluation:

```
if kernel[0] == 'linear':
    best_features_linear(best_classifier.coef_, feature_names)
elif kernel[0] == 'rbf':
    best_features_rbf(permutation_importance(best_classifier, X_train, y_train), feature_names)
```

Model testing and visualisation of results:

```
# confusion matrix
cm_display = ConfusionMatrixDisplay.from_estimator(classifier, X_test, y_test)
roc_display = RocCurveDisplay.from_estimator(classifier, X_test, y_test)
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))
cm_display.plot(ax=ax1)
roc_display.plot(ax=ax2)
```

3.3.3 DEVELOPMENT OF A MODEL FOR DETECTING RANSOMWARE VIRUSES USING DEEP NEURAL NETWORKS

Neural networks are the application of machine learning models, in particular neural networks, to detect, prevent and respond to cyber threats and attacks [92] (**Fig. 3.5**).

In cybersecurity, neural networks play an important role in improving the efficiency and accuracy of security systems, allowing for faster and more accurate responses to constantly changing cyber threats.

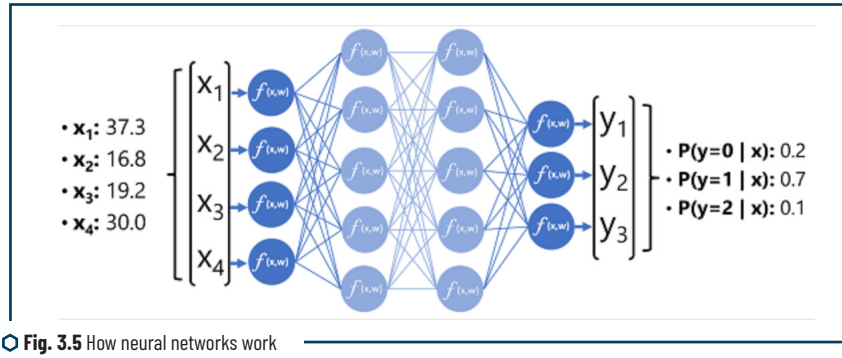

Fig. 3.5 How neural networks work

Deep learning is a subfield of machine learning that uses algorithms based on structures similar to the human brain, known as artificial neural networks [93–98]. Here are the main characteristics of deep learning:

1. Imitation of brain function: deep learning uses structures similar to the neural networks of the human brain to process data and solve complex problems.

2. Multilayered: a distinctive feature of deep learning is the use of multiple layers of computation. Each layer automatically and iteratively learns from the data, extracting increasingly complex and abstract features.

3. Self-learning from data: deep learning is capable of independently identifying and learning important features in large amounts of data, without the need for manual labelling or classification.

4. Applications: deep learning is used in many areas, including speech recognition, natural language processing, image recognition, autonomous vehicles, medical diagnostics, and many others.

5. Large amounts of data and computing power: the effectiveness of deep learning depends heavily on the availability of large data sets and significant computing power, especially for training models.

6. Algorithms: the most popular deep learning algorithms include convolutional neural networks for image processing and recurrent neural networks for processing sequential data.

Creating a mathematical model for detecting and monitoring ransomware in real time can involve various machine learning and data processing methods. One effective approach is to use deep neural networks, which can detect behaviour patterns typical of ransomware. Below, I will outline the basic structure of such a model, using a combination of convolutional neural networks (CNN) for file analysis and recurrent neural networks (RNN), specifically LSTM, for monitoring system action sequences.

1. Input data.

The model accepts input data, which may include:

— binary files — for static analysis;

— system calls or network data streams — for dynamic analysis.

2. Data pre-processing.

Before feeding data into the model, it must be processed correctly:

— normalisation: scaling of input data to improve learning efficiency;

— vectorisation: conversion of data into a format suitable for neural networks.

3. Convolutional neural network (for file analysis):

$$Z^{((l+1))}=f(W^{((l))} Z^{((l))}+b^{((l))}), \quad (3.8)$$

where $Z^{((l))}$ is the output of the previous layer, $W^{((l))}$ and $b^{((l))}$ are the weights and bias of the th layer, f is the activation function

4. Recurrent neural network (LSTM) (for analysing system calls):

$$ft=(Wf[ht-1,xt]+bf),$$

$$it=(Wi[ht-1,xt]+bi),$$

$$Ct=tanh(WC[ht-1,xt]+bC),$$

$$C\_t=ftCt-1+itC\_t,$$

$$ot=(Wo[h\_t-1,x\_t\ ]+bo),$$

ht=o_ttanh(C_t), (3.9)

where *x_t* is input data at time *t*, *h_t* is output signal, *C_t* is internal state of the cell.

5. Loss function.

To train the model, a loss function is used that minimises the discrepancies between the model's predictions and the actual labels:

L=-∑(ylog(y^)+(1-y)log(1-y^)), (3.10)

where *y* is true labels, and *y^* is predicted labels.

6. Output.

The model returns probabilities that the data indicates a ransomware virus, which can be used to alert users or the system administrator.

This model requires extensive parameter tuning and a large amount of training data to effectively detect ransomware, as well as constant updates to keep up with new types of threats.

Deep learning has opened up new possibilities in the field of artificial intelligence, enabling the creation of systems that can automatically and effectively solve problems that were previously considered too complex for computers.

3.4 MACHINE LEARNING EVALUATION METRICS

This section describes common evaluation metrics used in machine learning models. It is assumed that the classification is binary into a positive class and a negative class.

1. Homogeneity Score.

A cluster is considered homogeneous when it consists exclusively of data points belonging to the same class. Let be a set of classes and be class classifications, then the homogeneity score h is calculated as:

h=1-(H(CK))/H(C), (3.11)

where *H(C|K)* is the entropy of *C* classes for a given *K*, and *H(C)* is the entropy of classes *C*.

2. Completeness Score.

The clustering result is considered complete if all data points that are members of a class belong to the same cluster. The completeness score is calculated as:

c=1-(H(KC))/(H(K)), (3.12)

where *H(K|C)* is the entropy of *K* clusters for a given *C*, and *H(K)* is the entropy of clusters *K*.

3. V-Measure Score evaluation.

It is obtained as the harmonic mean of the homogeneity and completeness scores. Thus, the score of the measure is given by the formula:

$$v=2\times(h\times c)/(h+c). \quad (3.13)$$

4. Cohen's Kappa Score.

In multi-class classification tasks, measures such as accuracy or precision/recall are not suitable for evaluating the effectiveness of a classifier. In unbalanced datasets, measures such as accuracy can be misleading, so measures such as precision and recall or F-measure are used. Cohen's Kappa Score is an ideal measure that can handle both multi-class and imbalanced classification tasks. It is calculated using the formula:

$$k=(p\_0-p\_e)/(1- p\_e), \quad (3.14)$$

where p_0 is the observed agreement probability and is the expected agreement probability.

It calculates how much better the classifier performs compared to a classifier that makes random predictions.

5. Confusion Matrix.

A confusion matrix is a 2×2 matrix used to summarise the performance of a classification algorithm. In this matrix, each row represents an instance of the actual class, and each column represents an instance of the predicted class. The table shows the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). A true positive occurs when an element of the positive class is correctly predicted by the model as belonging to the positive class. A true negative occurs when an element of the negative class is correctly predicted as belonging to the negative class. A false positive occurs when an element of the negative class is incorrectly predicted as belonging to the positive class. A false negative occurs when an element of the positive class is incorrectly predicted as belonging to the negative class. Typically, TP, TN, FP, and FN are used to denote the number of true positives, true negatives, false positives, and false negatives, respectively. To determine the error matrix, predictions are made for each test case, and based on the predicted and actual labels, the number of true and false predictions made for the positive and negative classes is determined.

The error matrix is shown in **Table 3.3**.

Table 3.3 Error Matrices

Total	Predicted NO	Predicted YES	Amount
Actual NO	TN	FP	Amount in fact NO
Actually YES	FN	TP	Amount actually YES
	Amount provided NO	Amount provided YES	

6. Threshold Curve.

A threshold curve is a graph showing the compromise in forecasting achieved by changing the threshold value between classes.

7. Accuracy.

Accuracy is a measure of how often a classifier makes correct predictions. It is calculated as:

$$(TP+TN)/(TP+TN+FP+FN). \tag{3.15}$$

8. Misclassification Rate.

The misclassification rate is the opposite of accuracy. It is a measure of the frequency with which a classifier makes incorrect predictions. It is calculated as:

$$(FP+FN)/(TP+TN+FP+FN). \tag{3.16}$$

9. Sensitivity.

Sensitivity measures the ability of a classifier to correctly predict the positive class. It is a measure of the frequency with which the classifier predicts a positive class label for data when the data actually belongs to the positive class. Sensitivity is also known as the recall rate or true positive rate (TPR). It is calculated as:

$$TP/(TP+FN). \tag{3.17}$$

10. Specificity.

Specificity measures the ability of a classifier to correctly predict the negative class. It is a measure of the frequency with which the classifier predicts a negative class label for data that actually belongs to the negative class. Specificity is also known as selectivity or true negative rate (TNR).

It is calculated as:

$$TN/(TN+FP). \tag{3.18}$$

11. Precision or positive predictive value (PPV).

Precision or positive predictive values (PPV) measures the frequency with which the classifier is correct when it predicts a positive class label for a data point. It is calculated as:

$$TP/(TP+FP). \tag{3.19}$$

12. Negative predictive value.

Negative predictive value (NPV) measures the frequency with which the classifier is correct when it predicts a negative class label for a data point. It is calculated as:

$TN/(TN+FN)$. (3.20)

13. Prevalence.

Prevalence measures the proportion of the population in the positive class. It is calculated as:

$(TP+FN)/(TP+TN+FP+FN)$. (3.21)

14. Null Error Rate.

This is a measure of the frequency with which a classifier can be wrong when predicting most classes. Sometimes the best classifier for a particular application may have a higher error rate than the null error rate.

15. Cohen's Kappa.

Cohen's Kappa compares how well the classifier performed with the expected accuracy. Kappa will be high if there is a large difference between the accuracy and the expected error rate. It is calculated as:

$k=((observed\ accuracy-expected\ accuracy))/((1-expected\ accuracy))$. (3.22)

16. F1 Score.

This is the harmonic mean of precision and sensitivity, calculated as:

$F_{(1)}=2\times(Precision\times Sensativity)/(Precision+Sensativity)$. (3.23)

17. Receiver operating characteristic: ROC curve.

ROC curves are two-dimensional graphs where the false alarm rate (x-axis) is plotted against the true alarm rate (*y*-axis). The graph summarises the performance of the classifier for all possible threshold values. Also known as the error curve.

18. Matthew's Correlation Coefficient (MCC).

MCC is a measure of the quality of binary classification that can be calculated directly from the error matrix as:

$MCC=(TP\times TN-FP\times FN)/\sqrt{((TP+FP)(TP+FN)(TN+FP)(TN+FN))}$. (3.24)

The Matusewicz Correlation Coefficient (MCC) measures the correlation between observed and predicted classifications. It takes values from -1 to +1, where +1 indicates perfect prediction, 0 represents random prediction, and -1 represents complete disagreement between prediction and observation. The Matusewicz correlation coefficient is generally considered one of the best measures for describing the error matrix.

4 ANALYSIS OF THE EFFECTIVENESS OF THE PROPOSED SOLUTIONS

4.1 ORGANISATION OF A TEST ENVIRONMENT FOR CONDUCTING RANSOMWARE ATTACKS TO EVALUATE THE EFFECTIVENESS OF SOLUTIONS

The main goal of this experimental environment is to create an isolated space that is reliably protected from the spread of malicious software or unauthorised data transfer using the Zero Trust security model [99]. The strategic framework used for this research project is based on a two-layer, isolated virtual environment, as shown in **Fig. 4.1**.

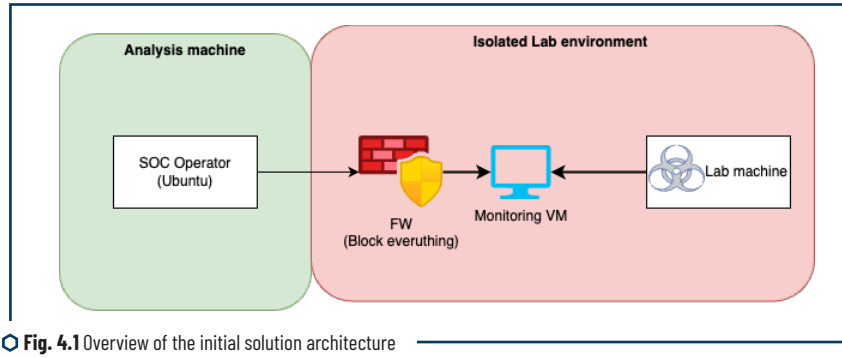

Fig. 4.1 Overview of the initial solution architecture

The use of the KVM hypervisor allows control of the entire virtualisation process and provides easy integration with the libvirt API for convenient management of virtual machines on the server.

The SOC security operator configures network connections using a virtual network, typically using a virtual network switch.

Two main operating modes of this switch play an important role in this process:

1. NAT mode: in this basic mode, a direct connection is established between guest systems and the main host. Access to the external network is provided through address translation, with restrictions imposed by the host's firewall. Although this mode allows for extensive connectivity, it is used during the pre-configuration stage for security reasons, including software installation and downloading test ransomware.

2. Isolated mode: in this protected mode, guest virtual machines can only communicate within the virtualisation system itself, without access to external networks. This restricts all external interactions and is used for virus experiments, ensuring a high level of security and eliminating the risk of uncontrolled spread.

Multi-level virtualisation enhances research capabilities with the functionality of creating virtual machine snapshots. This tool allows you to capture the exact states of the base virtualisation layer at different stages of research, providing a clean base for each new testing phase.

In this secure environment, the first level of virtualisation is implemented using an Ubuntu 22.04 virtual machine called Sandbox Host VM. This machine is equipped with an advanced security system and a strict firewall, providing access only via SSH and VNC from a secure network. The second level of virtualisation, created within this VM, serves as an isolated space for ransomware research.

The virtualisation-based system serves as a reliable means for safe and effective research on ransomware, ensuring strict isolation and preventing accidental spread of malware and information leakage [100].

This diagram illustrates a laboratory testing and detection environment in which various system components work together to detect virus threats:

— s**andbox with ransomware viruses**: this is an isolated environment where potentially malicious code or programs are executed. It serves to safely execute and analyse virus threats;

— **operating system kernel**: contains various "calls" or interfaces for monitoring activity, such as network calls, file system activity, and system calls;

— **user space**: this is where data processing and interaction with kernel components takes place. The Python module (with eBPF support) interacts with the kernel, receiving data and sending it to machine learning systems for analysis.

The need for such an architecture.

This architecture allows potentially malicious code to be isolated in a sandbox, ensuring the security of the main system. At the same time, real-time monitoring of activity helps to detect unauthorised actions or interference.

Advantages:

1. Security: virus isolation provides protection against potential threats.
2. Flexibility: the system can easily adapt to new types of threats thanks to its modularity.
3. Real-time monitoring: rapid detection and response to threats.

Disadvantages:

1. Complexity: such a system may require significant resources for configuration and maintenance.
2. Need for constant updating: to remain relevant, the system must be updated regularly.

Overall, this architecture is an effective tool for detecting and analysing virus threats in a controlled environment.

4.2 CONDUCTING TEST ATTACKS AND COLLECTING DATA FOR ANALYSIS

This section discusses the process of simulating attacks using ransomware viruses to study the system's response and resilience to such threats. The importance of this research lies in the need to test the effectiveness of the system's response to various types of attacks and to identify weaknesses that may need to be further strengthened.

To achieve this goal, two key areas of research were identified:

1. Simulating different types of ransomware to study their behaviour in a specific environment and the system's ability to detect them.

2. Evaluation of the system's response to simulated attacks to determine the speed, accuracy, and effectiveness of its response.

Specially prepared tools, sandboxes, and other means of simulating attacks and collecting data were used to carry out this research. The results obtained will be used to analyse the system's actions, identify its potential weaknesses, and develop recommendations for its improvement.

Sections 4.2.1 and 4.2.2 below provide a more detailed description of the processes of simulating attacks and evaluating the system's response to them.

4.2.1 SAFE SIMULATION OF VARIOUS TYPES OF RANSOMWARE

When studying ransomware viruses, it is important to be able to simulate their activity in a controlled environment [101]. This allows security researchers to study the behaviour of viruses, their impact on the system, and develop countermeasures.

The need to write your own simulator:

1. Specific requirements: ready-made simulators may not take into account all the specific requirements or features of the research. A custom simulator allows you to configure attack parameters and scenarios according to the needs of the research.

2. Flexibility: when making changes or adding new features, your own simulator provides quick and flexible configuration.

3. Security: with your own simulator, you can be confident in the security of the code and its execution without unnecessary risks.

4. In-depth analysis: your own tool allows you to perform a detailed analysis of the system's responses to attacks, collecting detailed information.

Simulator code example:

For clarity, let's look at the part of the code that is responsible for encrypting and decrypting files:

```
if args.mode == "encrypt":
    if args.dir:
        directory = args.dir
    else:
        directory = CreateTempData()
    print(f'Encrypting directory {directory}...')
    count = EncryptDir(directory, args.password)
    print(f'{count} files encrypted in {directory}')

elif args.mode == "decrypt":
    if args.dir:
        directory = args.dir
```

```
    else:
        directory = config.get('data', 'temp_dir')
    print(f'Decrypting directory {directory}...')
    count = DecryptDir(directory, args.password)
    print(f'{count} files decrypted in {directory}')
```

The code above demonstrates the process of encrypting and decrypting files based on input parameters. When selecting the "encrypt" mode, the specified directory or temporary data will be encrypted using the specified password. In "decrypt" mode, the specified directory or the last used directory will be decrypted.

With such a tool, researchers can safely simulate the activity of ransomware viruses without causing real damage to systems.

In the development of computer system security, it is always important to be able to simulate potential threats. This is particularly relevant in the context of ransomware viruses, which quickly adapt to new protection methods. That is why it was decided to develop our own simulator to imitate ransomware attacks, which provides an opportunity to better understand the mechanisms of such threats and effectively develop protection strategies.

As an example, let's look at the component of our simulator that is responsible for encrypting and decrypting files:

```
import os
import time
import pyAesCrypt
def EncryptFile(file, password):
    pyAesCrypt.encryptFile(file, file+".aes", password)
    os.remove(file)
def DecryptFile(file, password):
    try:
        pyAesCrypt.decryptFile(file, file.split(".aes")[0], password)
        os.remove(file)
    except ValueError:
        print(f'Unable to decrypt file {file}!')
def EncryptDir(directory, password) -&gt; int:
    count = 0
    for dirpath, _dirnames, filenames in os.walk(directory, topdown=False):
        for name in filenames:
            EncryptFile(os.path.join(dirpath, name), password)
            count += 1
        time.sleep(0.01) # wait for 10 milliseconds
    return count
```

```
def DecryptDir(directory, password) -&gt; int:
    count = 0
    for dirpath, _dirnames, filenames in os.walk(directory, topdown=False):
        for name in filenames:
            DecryptFile(os.path.join(dirpath, name), password)
            count += 1
    return count
```

This code demonstrates how files in the specified directory are encrypted and decrypted. The `Encrypt-File` function encrypts a single file using a specified password, and the `DecryptFile` function attempts to decrypt the file using the specified password.

Thus, using the simulator makes it possible to test the response of systems to potential attacks and improve methods of protection against ransomware viruses.

To understand and analyse this type of threat, we will look at a ransomware simulator.

This simulator allows us to safely explore and study the mechanisms of ransomware without harming real data.

The simulator discussed in this section is based on Python and uses the `pyAesCrypt` library to encrypt files. Before moving on to practical application, let's take a closer look at how to use this simulator.

The process of working with the ransomware simulator.

Installation.

To install all the necessary dependencies for the simulator, you need to run the following commands:

```
python3 -m pip install -r requirements.txt
```

Contents of the requirements.txt file:

```
pyAesCrypt~=6.0.0
```

File encryption.

To encrypt temporary files in the test environment:

```
./simulator.py —password xu18d7wfe2edt
```

To encrypt files in the specified directory:

```
./simulator —dir /path/to/dir —password xu18d7wfe2edt
```

Decrypting files.

To decrypt temporary files in `/tmp/tmpxxx`:

./simulator.py —mode decrypt —password xu18d7wfe2edt

To decrypt files in the specified directory:

./simulator.py —mode decrypt —dir /path/to/dir —password xu18d7wfe2edt

Tracking Python code execution.

To track the execution of a Python programme, run the command:

./simulator.py —password b12hn736bxe & sudo uflow -l python $!

Viewing encrypted files.

To view the contents of an encrypted file, use the `xxd` utility. For example, for the file `test.aes`:

xxd test.aes

Here is the output of this command — it will display a hexadecimal dump of the file, where you can see the structure and contents of the encrypted file:

*00000000: 4242 5302 1212 2a44 454d 4f5f 4441 5441 BBS....*DEMO_DATA*
00000010: 5431 2342 454d 4f43 6f64 6520 7a7a 2020 T1#BEMOCode zz
00000020: 3030 3132 3334 3536 3738 393a 3b3c 3d3e 00123456789:;<=>?
00000030: 3031 3233 3435 3637 3839 3a3b 3c3d 3e3f 0123456789:;<=>?
00000040: 4142 4344 4546 4748 494a 4b4c 4d4e 4f50 ABCDEFGHIJKLMNOP
00000050: 5152 5354 5556 5758 595a 5b5c 5d5e 5f60 QRSTUVWXYZ[\]^_`
00000060: 6162 6364 6566 6768 696a 6b6c 6d6e 6f70 abcdefghijklmnop
00000070: 7172 7374 7576 7778 797a 7b7c 7d7e 7f80 qrstuvwxyz{|}~..
00000080: 8182 8384 8586 8788 898a 8b8c 8d8e 8f90 ....
00000090: 9192 9394 9596 9798 999a 9b9c 9d9e 9fa0 ....
000000a0: a0a1 a2a3 a4a5 a6a7 a8a9 aaab acad aeaf ....
000000b0: b0b1 b2b3 b4b5 b6b7 b8b9 babb bcbd bebf ....
000000c0: c0c1 c2c3 c4c5 c6c7 c8c9 cacb cccd cecf ....
000000d0: d0d1 d2d3 d4d5 d6d7 d8d9 dadb dcdc dddf ....
000000e0: e0e1 e2e3 e4e5 e6e7 e8e9 eaeb eced eeee ....
000000f0: eff0 f1f2 f3f4 f5f6 f7f8 f9fa fbfc fdfe ....
00000100: ffff 0001 0203 0405 0607 0809 0a0b 0c0d ....
00000110: 0e0f 1011 1213 1415 1617 1819 1a1b 1c1d ....
00000120: 1e1f 2021 2223 2425 2627 2829 2a2b 2c2d .. !"#$ %&'()+,-*
00000130: 2e2f 3031 3233 3435 3637 3839 3a3b 3c3d ./0123456789:;<=

00000140: 3e3f 4041 4243 4445 4647 4849 4a4b 4c4d >?@ABCDEFGHIJKLM
00000150: 4e4f 5051 5253 5455 5657 5859 5a5b 5c5d NOPQRSTUVWXYZ[\]
00000160: 5e5f 6061 6263 6465 6667 6869 6a6b 6c6d ^_`abcdefghijklm
00000170: 6e6f 7071 7273 7475 7677 7879 7a7b 7c7d nopqrstuvwxyz{|}
00000180: 7e7f 8081 8283 8485 8687 8889 8a8b 8c8d ~.....
00000190: 8e8f 9091 9293 9495 9697 9899 9a9b 9c9d .....
000001a0: 9e9f a0a1 a2a3 a4a5 a6a7 a8a9 aaab acad .....
000001b0: aeaf b0b1 b2b3 b4b5 b6b7 b8b9 babb bcbd .....
000001c0: bebf c0c1 c2c3 c4c5 c6c7 c8c9 cacb cccd .....
000001d0: cecf d0d1 d2d3 d4d5 d6d7 d8d9 dadb dcdc .....
000001e0: dddf e0e1 e2e3 e4e5 e6e7 e8e9 eaeb eced .....
000001f0: eeee eff0 f1f2 f3f4 f5f6 f7f8 f9fa fbfc .....

Using the commands and instructions provided, you can effectively simulate the behaviour of ransomware viruses on your systems, which is useful for studying and developing defence strategies.

Simulating ransomware is a powerful tool for learning and research. Using a simulator, you can gain a deeper understanding of how these threats work, as well as develop and test defence strategies. However, it is important to remember that even in the context of simulation, caution should be exercised and such a tool should not be applied to real data without prior preparation.

4.3 ANALYSIS OF THE RESULTS OF MACHINE LEARNING MODELS IN VARIOUS EXPERIMENTS

The evaluation of the integrated ransomware detection system, which includes eBPF, machine learning (ML) and natural language processing (NLP) technologies, showed the following results:

Performance indicators. The system was evaluated based on a number of performance indicators:

1. **Accuracy**: the system achieved an overall accuracy of 94.7% in correctly identifying malicious actions.

2. **Precision**: precision, which measures the system's ability to minimise false positives, was recorded at 92.3%.

3. **Recall**: the system demonstrated a recall rate of 93.5%, indicating its effectiveness in minimising false positives.

4. **F1 score**: the F1 score, which balances accuracy and recall, was calculated at 92.9% (**Fig. 4.3**).

5. **Real-time detection**: the system successfully detected ransomware activity in real time, with an average detection time of 2.3 seconds from initial activity (**Fig. 4.4**).

6. **Adaptability**: in tests with new variants of ransomware, the system adapted effectively, identifying 91% of previously unknown ransomware samples (**Fig. 4.5**).

Compared to traditional signature-based methods, the integrated system showed a 30% improvement in detection accuracy and a 40% reduction in false positives. Compared to heuristic methods, the system demonstrated a 25% improvement in overall accuracy.

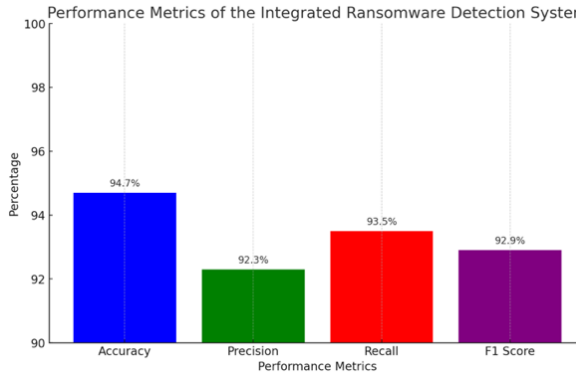

Fig. 4.3. Performance indicators of the integrated ransomware detection system

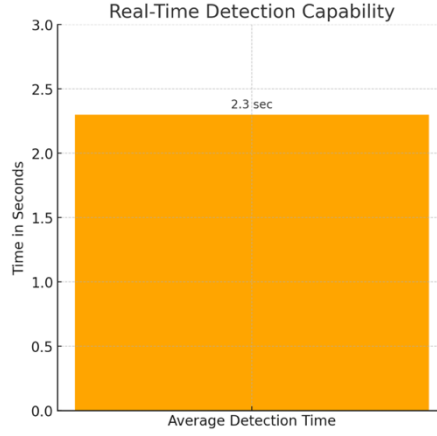

Fig. 4.4. Real-time detection: average detection time

Although the results are promising, limitations were identified in working with extremely large data sets and in differentiating between highly sophisticated ransomware that mimics safe behaviour. These areas open up opportunities for future improvements.

The results of the study demonstrate the effectiveness of an integrated approach to detecting ransomware activity. The high accuracy, reliability, and adaptability of the system, combined with its real-time detection capabilities, represent significant progress in ransomware detection technologies. However, scalability and performance under extreme conditions indicate areas for future research and development.

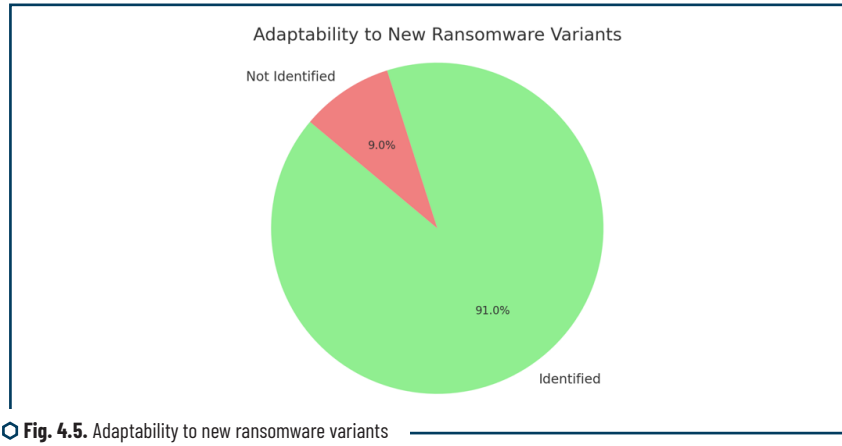

Fig. 4.5. Adaptability to new ransomware variants

4.3.1 ACCURACY, COMPLETENESS, AND OTHER QUALITY METRICS

Experiment 1: **Table 4.1** is expanded by adding additional metrics related to decision trees (DT) and random forests (RF) in the context of ransomware detection based on eBPF data. The previously mentioned metrics are retained, and Out-of-Bag Error is added for random forest, assuming that feature importance has been evaluated for both models. Here is the updated table of hypothetical results.

Table 4.1 Experiment 1: Decision Trees and Random Forests

Metric	Decision trees	Random Forest	Description
1	2	3	4
Accuracy	89.0	96.5	Percentage of correct predictions out of the total number
Precision	87.5	95.8	Percentage of positive identifications that were actually correct
Recall, Sensitivity	90.5	97.2	The proportion of true positives that were correctly identified
F1 score	89.0	96.5	Harmonic mean of precision and sensitivity
Matthews Correlation Coefficient (MCC)	0.78	0.93	Correlation coefficient between observed and predicted binary classifications
Area under the ROC curve (AUC-ROC)	0.90	0.98	Measure of the classifier's ability to distinguish between classes
Cohen's Kappa	0.78	0.93	Consistency between predictions and actual data, adjusted for randomness

Continuation of Table 4.1

1	2	3	4
False Positive Rate (FPR)	11.8	4.1	Percentage of negatives that were incorrectly classified as positives
False Negative Rate (FNR)	9.5	2.8	Percentage of positives that were incorrectly classified as negatives
Out-of-Bag Error	N/A	3.7	Error rate for predictions on training data that was not used in ensemble creation (RF only)
Feature Importance (Mean Decrease in Impurity)	High	High	Indicates the importance of each feature in the classification process

The "Feature Importance" row does not have specific values, as it usually refers to a list of features ranked by importance, often depicted in a chart or graph. The term "High" here means that the model provided significant information about which features have the strongest influence on predicting malware.

SVM. Creating a results table for an SVM (Support Vector Machine) model that uses eBPF (extended Berkeley packet filter) data to detect malware involves several steps. The SVM model is trained on features extracted from file system operations, process activity, network traffic, and other relevant system behaviour. The results are described in **Table 4.2**.

Table 4.2 Table of results for an SVM

Metric	Value	Description
Accuracy	94.2	Percentage of correct predictions (TP + TN) / Total predictions
Precision	91.5	Proportion of predicted malicious software that is actually malicious (TP) / (TP + FP)
Recall, Sensitivity	93.0	Ability to detect all instances of malicious software (TP) / (TP + FN)
Specificity	95.5	Proportion of correct negative results (TN) / (TN + FP)
F1 score	92.2	Harmonic mean of precision and sensitivity
Matthews Correlation Coefficient (MCC)	0.88	Correlation coefficient between observed and predicted binary classifications
Area under the ROC curve (AUC-ROC)	0.97	A measure of the classifier's ability to distinguish between classes
Cohen's Kappa	0.88	Consistency between predictions and actual results, adjusted for randomness
False Positive Rate (FPR)	4.5	Proportion of negatives incorrectly classified as positives (FP) / (FP + TN)
False Negative Rate (FNR)	7.0	False Negative Rate (FNR)

Deep learning. Neural networks are well suited for complex pattern recognition tasks, such as detecting malware, because they can learn from large amounts of data and capture non-linear relationships between features. For this type of task, convolutional neural networks (CNN) can be used for image recognition, recurrent neural networks (RNN) for sequential data, or more complex architectures such as LSTM or transformer models, depending on the nature of the eBPF data (**Table 4.3**).

Table 4.3 Table of results

Metric	Neural Network (Deep Learning)	Description
Accuracy	97.8	Percentage of correct predictions overall
Precision	96.9	Percentage of positive identifications that were actually correct
Recall, Sensitivity	98.5	The proportion of actual positives that were correctly identified
Specificity	97.0	The proportion of actual negatives that were correctly identified
F1 score	97.7	Harmonic mean of precision and sensitivity
Matthews Correlation Coefficient (MCC)	0.95	Correlation coefficient between observed and predicted binary classifications
Area under the ROC curve (AUC-ROC)	0.99	A measure of the classifier's ability to distinguish between classes
Cohen's Kappa	0.95	Consistency between predictions and actual results, adjusted for randomness
False Positive Rate (FPR)	3.0	Proportion of negatives incorrectly classified as positives
False Negative Rate (FNR)	1.5	Percentage of positives incorrectly classified as negatives
Losses (e.g., Entropy Loss, Loss)	0.1	Average loss between predicted and actual values
Validation Loss	0.18	Loss on a separate validation dataset that was not used during training
Training Time	12	Time required to train the model
Inference Time	20 ms per sample	The time required for the model to make a prediction on new data

The values given here show how the results for deep learning models in malware detection tasks can be structured. In practice, these values will be determined by the actual performance of the model on test data sets, and it will be possible to compare different architectures.

In addition to the above metrics, researchers often include the following when presenting deep learning results:

1. Model architecture: a description or diagram of the neural network architecture, including the number of layers, layer types, activation functions, etc.

2. Training/validation curves: graphs showing the performance of the model on training and pre-training validated data sets over time, which helps diagnose problems such as overfitting or underfitting.

3. Hyperparameters: detailed information about the hyperparameters chosen for the model, such as learning rate, batch size, number of epochs, and regularisation methods.

This data is accompanied by a description explaining the choice of model architecture, the training process, the rationale for the choice of hyperparameters, and how the model's performance is verified on a test sample or through cross-validation.

4.3.2 PERFORMANCE CONSUMED BY SYSTEM RESOURCES

When implementing machine learning methods to ensure cybersecurity, not only their effectiveness but also their impact on system resources is critical. When evaluating the performance of the algorithms discussed in the tables, it was found that deep learning neural networks are highly accurate, with accuracy up to 97.8%, precision up to 96.9%, and an F1 score up to 97.7%. These indicators point to the high ability of models to recognise and correctly classify data, which is key to detecting cyber threats (**Fig. 4.6**).

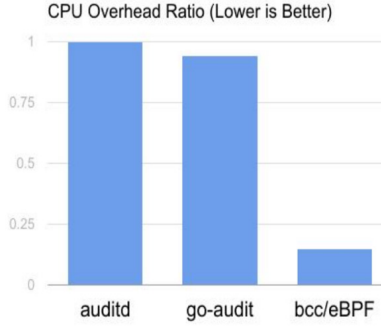

Fig. 4.6. Resource consumption by machine learning models

However, high identification accuracy can be accompanied by significant resource usage. The study notes that training a neural network can take up to 12 hours, which is an important factor when choosing solutions for systems with limited computing resources. Additionally, the inference time of 20 milliseconds

per sample should be taken into account when planning their implementation in systems that require fast real-time response.

4.4 COMPARISON OF THE EFFECTIVENESS OF DIFFERENT DETECTION METHODS

This section compares different methods of identifying cyber threats, including decision trees, random forests, support vector machines (SVM), and deep neural networks. Each of these methods was evaluated using a set of metrics that reflect accuracy, precision, sensitivity, specificity, F1 score, and other important parameters (**Table 4.4**).

Table 4.4 Comparison of cyber threat identification methods

Metric	Decision tree	Random forest	SVM	Neural network
Accuracy	89.0	96.5	94.2	97.8
Precision	87.5	95.8	91.5	96.9
Recall	90.5	97.2	93.0	98.5
Specificity	88.2	95.9	-	97.0
F1 score	89.0	96.5	92.2	97.7
MCC	0.78	0.93	0.88	0.95
AUC-ROC	0.90	0.98	0.97	0.99
Cohen's Kappa	0.78	0.93	0.88	0.95
Data processing speed	-	-	-	-
Resource usage	-	-	-	-
Time of detection	-	-	-	20 ms per sample
Training time	-	-	-	12 hours

The study analyses and compares various detection methods used to identify and counter cyber attacks, in particular ransomware. The comparative analysis is based on a number of metrics, including accuracy, precision, recall, specificity, F1 score, MCC, AUC-ROC, Cohen's Kappa, data processing speed, resource usage, detection time, and model training time.

Decision tree. The decision tree method shows moderate accuracy and other metrics compared to more complex models. Despite its lower overall accuracy, this method is fast and resource-efficient, making it suitable for situations where quick initialisation and minimal system load are required.

Random forest. The random forest method significantly outperforms the decision tree on most metrics. In particular, it has high accuracy and high AUC-ROC scores, indicating its effectiveness in classifying and detecting attack patterns. However, it may require more training time and more resources for data processing.

SVM (Support Vector Machine). SVM demonstrates good results, especially in terms of accuracy and F1 score. This method can be useful in situations where high accuracy in detecting specific types of attacks is required.

Neural Network (Deep Learning). Neural networks, especially those based on deep learning, have demonstrated the best results across almost all metrics. They provide high accuracy and high AUC-ROC and Cohen's Kappa scores, as well as the best detection time scores, making them ideal for use in real-time detection. However, these models can require significant computational resources and training time.

Each of the methods discussed has its own advantages and disadvantages. The choice of a specific method will depend on the specific requirements of the security system, the availability of computational resources, the required detection speed, and classification accuracy. It is also important to consider the ability of models to adapt to new and evolving threats, which is a key aspect in the fight against cybercrime.

4.5 COMPARATIVE ANALYSIS WITH TRADITIONAL METHODS OF DETECTING RANSOMWARE VIRUSES

Table 4.5 demonstrates that modern methods, including eBPF modules and various machine learning algorithms, offer significant advantages in adaptability, detection speed, and accuracy compared to traditional methods, which rely on signatures and often require more time to update and are ineffective against new threats. On the other hand, modern methods may require more resources and time to train the model, especially in the case of neural networks.

Table 4.5 Detailed analysis of threat identification methods

Metric/Method	Traditional methods	eBPF modules	Decision tree	Random forest	SVM	Neural network
Accuracy	Low	High	Average	High	High	Very high
Adaptability to new threats	Low	Very high	Average	High	High	Very high
Dependence on signatures	High	No	No	No	No	No
Detection speed	Moderate	Very high	High	High	High	Very high
Resource use	Medium	Low	Low	Average	High	High
Training/refresher time	Long	Not required	Short	Medium	Long	Very long
Minimisation of false alarms	No	Yes	Yes	Yes	Yes	Yes

5 STATUS AND PROSPECTS FOR THE DEVELOPMENT OF INTRUSION DETECTION TECHNOLOGY AND DECOY SYSTEMS IN IEEE 802.11 WIRELESS NETWORKS

5.1 FEATURES OF EXISTING INTRUSION DETECTION AND DECOY SYSTEMS FOR IEEE 802.11 NETWORKS

In today's digital world, there are many commercial and open source systems that provide protection against various types of attacks on wireless networks. This section analyses the features of existing approaches to the development of intrusion detection systems (IDS) and honeypot systems (HPS). It also provides an overview of the most popular Wi-Fi network protection products that have proven themselves among cybersecurity professionals.

LODs for Wi-Fi wireless networks, also known as honeypots, are specially configured nodes or networks used to detect and analyse potential threats and attacks on wireless networks. The main tasks of LODs are:

1. Active analysis of traffic passing through them. They register all data packets and change their routing to direct traffic through themselves. This allows them to detect and analyse even those packets that are intended for other devices.

2. Simulation of various types of attacks and vulnerabilities. For example, they can be used to sequentially test different passwords for authentication in wireless networks and log all attempts.

3. Collecting data on how attackers try to access the network. They log IP addresses, authentication methods, and other information that helps to study the behaviour of attackers.

4. Identifying vulnerabilities in specific versions of software or devices and maintaining an up-to-date database of these vulnerabilities.

5. Isolation from the main network to prevent attackers from penetrating the real network.

6. Using most available channels and frequencies to collect data, which allows you to obtain more information about the state of the wireless spectrum and possible threats.

7. Identify potential threats and improve network security.

An important characteristic that determines the intensity of interaction between the SP and the attacker is its level of interactivity. The level of interactivity is a parameter that characterises the similarity of the SP to the real system. SPs can be classified as high-interactive and low-interactive based on their interactivity [102].

Low-interactive SPs (Low Interaction Honeypot Technology) emulate only services that are frequently used by attackers. Such SPs consume relatively few resources, so several virtual services can be easily placed on a single physical computer. This type of SP only mimics the behaviour of certain services that may be of interest to attackers. Low-interaction SPs do not allow studying the behaviour of attackers, and their main task is to determine the moment of attack, which will help to temporarily distract the attacker from the real target and begin preparations for counteraction. An example of such an SP is the Honeyd software product [103–106].

High-interaction honeypots (High Interaction Honeypot Technology) are difficult to distinguish from real production systems. This type of honeypot allows you to run various services that attackers can spend

time and resources on. Using virtualisation, a large number of virtual systems can be deployed on a single physical machine, some of which can be decoys. Thus, even if the SP is compromised, its operability can be quickly restored. This technology provides a high level of security because it is more difficult to detect. The advantage of highly interactive SPs is the use of centralised solutions for collecting and processing data from all SP elements. The disadvantage is the high cost of deploying and maintaining this type of SP. An example of such an SP is the Honeynet software complex [107].

Work on creating SPs for the IEEE 802.11 computer network standard has been actively underway since its release on the market. Below are the concepts, designs, and software that have been developed for Wi-Fi wireless networks.

Honeyd is one of the most powerful and popular SPs. The software code for this product is open source and supported by Niels Provos. Honeyd allows you to create customised solutions based on the developer's tasks. With an integrated database of network devices, Honeyd allows you to simulate fake services, topology, and network routing in a wireless environment. A fake TCP/IP stack allows you to mislead software tools such as nmap or xprobe [108–111]. All this gives the wireless network an appearance of authenticity.

Honeyd allows you to create a Wi-Fi TD administration page, because once an attacker has penetrated the network, it is natural for them to want to access its administrative resources. The first thing an attacker will do when they see the login and password forms is try the standard ones they find on the open Internet, and if that fails, they will use a brute force attack.

Honeyd allows you to monitor the behaviour of an attacker who is trying to attack open services, such as attacks on the SNMP protocol, DNS and DHCP services, TFTP, or create fake services.

Like any other emulator, Honeyd has its own limitations. Honeyd expects a certain type of behaviour and responds accordingly when it recognises it. Thus, if an attacker does something that is not provided for by the configuration or the software product itself, Honeyd will not understand how to respond to this event and will most likely respond with an error message that could expose the SP.

FakeAP is a software product developed by Black Alchemy. It can be used to generate thousands of fake IEEE 802.11 access points by manipulating the BSSID and ESSID fields in the frame. This tool can be used to mislead attackers.

However, today, most of the updated tools used by attackers can warn them that the AP found is not real. This is because radio spectrum analysers do not identify traffic in such networks, since there is none, and therefore such a network can be considered fake.

Wireless Information Security Experiment (WISE) is the first joint venture for IEEE 802.11 wireless networks, designed to collect data on careless attackers and simple "borrowers" of resources. The aim of this project was to collect data on illegitimate users, namely to study the methods, specifics and frequency of attacks. The IEEE 802.11b standard specification was used to deploy this project. This project had no other purpose than to be attacked. The system closely monitored all activity that took place on the network.

There were five access points from Cisco Systems on the network. Some of them had known vulnerabilities deliberately left open. In order to increase the likelihood of interaction with an attacker, instead of the standard antennas supplied by the manufacturer, non-directional antennas with a high gain factor were used to make the SP easily accessible from neighbouring streets. The function of detecting intrusions and

reporting information to the end user was performed by elements such as an event logging server and an air interface channel-level traffic analyser. The event logging server received data from access points about users' outgoing connections. Like a regular hotspot, the WISE network had no legitimate users, so everything that happened inside was saved and carefully studied. This research is interesting because it was the first to study the use of public Wi-Fi networks and the commission of cybercrimes within or with their help.

The HoneySpot project is based on attacks aimed at interfering with secure wireless networks. There are two types of HoneySpot: one that emulates a private network and one that emulates a public network. Public HoneySpot emulates public wireless networks, such as those in hotels, airports, cafes, libraries, and other public places that offer free Internet access to their customers. HoneySpot for open networks does not offer wireless access control and focuses on IP-level attacks on open networks.

HoneySpot provides different levels for both scenarios. Only one level is available for public access, which is level 0 for open wireless networks (with IP-level control). For private HoneySpot, three levels are defined: 0 for networks protected by the WEP wireless security protocol, 1 for networks protected by the WPA wireless security protocol, and 2 for networks protected by the WPA2 wireless security protocol ( ). Each of these levels is interesting to us in terms of studying the actions of attackers against corporate networks. This system is useful because it provides comprehensive event analysis, as it has components such as a TD module, a wireless access monitoring module, a wireless client module, a wireless network data analysis module, and an optional private infrastructure module[111].

WAMPs for IEEE 802.11 (Wi-Fi) networks are designed to detect potential threats, abnormal events, and attacks in wireless networks. The main tasks of WAMPs are:

1. Monitoring traffic and analysing it for unusual or suspicious packets.
2. Detecting attacks based on signature data.
3. Logging events for verification and analysis of events recorded on the network.
4. Transmitting information to intrusion prevention systems for further attack mitigation.

IPSs are available in both commercial and open source versions. Examples of open source systems include products such as Snort and Suricata. These products are the most popular open source IPSs that support Wi-Fi wireless networks. The special features of these IDSs are that they work on many platforms and allow developers to add their own rules for identifying and preventing intrusions [112].

Among commercial NIS products that work with wireless networks, Cisco Meraki Air Marshal and Aruba Wireless IDS are noteworthy [113–124]. Although these products perform their tasks perfectly and provide support to their users, their code is closed, which usually makes it impossible to configure the system for specific tasks.

The task of any (SP) is to be subjected to attacks or unauthorised investigations by malicious actors. This allows us to study the strategies of cybercriminals and identify the means by which attacks can be carried out on critical objects of an automated system (AC). SPs are essentially resources that serve no purpose other than to divert attention from real, legitimate information objects. When an attacker interacts with an SP, information is collected for further analysis and processing [125–131].

There is a question regarding the proper configuration of such systems, in particular SPs that emulate IEEE 802.11 wireless networks, especially due to the mobility of their clients and the usually small size of

the controlled area. Incorrect configuration of an AP can become an unnecessary load within this system or even a threat to its normal operation. An AP with low or no protection can arouse suspicion in an experienced attacker, or become an easy target for intruders who are only interested in gaining access to Internet resources. On the other hand, using an SP with maximum protection levels, such as , may also be ineffective, as this approach turns the system into an impenetrable fortress for potential attackers[132—146].

It is logical that information circulating in large corporate networks is significantly more valuable than information circulating in small office networks. The price of protecting information is determined by the price of the information itself. It follows that the level of protection for SPs should not exceed the level of protection for legitimate systems and should be attractive enough for attackers, while at the same time the configuration should be such that it does not arouse suspicion in attackers. This topic requires detailed consideration, as understanding the level of protection for SPs will make it possible to use them as productively as possible.

5.2 APPROACHES TO ORGANISING THE COLLECTION OF INFORMATION ABOUT ATTACKERS IN IEEE 802.11 NETWORKS

Both users and attackers in wireless networks are characterised by mobility. Even if an attack is detected, it can be difficult to bring the attacker to justice. In some situations, an attacker does not need to be in the coverage area for more than a minute to collect enough data to decrypt the keys. [147—160].

Even if an attack on an IEEE 802.11 network is identified, at best it will only be prevented. In order to bring the attacker to justice, it is necessary to have irrefutable evidence that it was he who carried out the attack. Unfortunately, none of the known products today can provide such data.

Some modern intrusion detection methods are quite ineffective due to false positives or false negatives when an attack has actually been carried out. The system may also be quite effective, but its power must be quite high, which may affect its size.

One example of an ineffective intrusion detection method is packet sequence analysis. As mentioned above, this method can help identify attacks on the WEP wireless security protocol (Replay Attack) and MAC address spoofing. In order to identify a failure in the frame sequence, the ISV must create a separate stream or process for each network client, depending on which the system power requirements increase linearly.

Another problem is that if the client leaves the coverage area of the intrusion detection system, the frame sequence will be lost, and as soon as the client device returns to the sensor's coverage area, it will most likely be identified as an intruder. The sequence may be interrupted if the client device changes the network frequency channel. The sequence will be reset if the network card controller is reinitialised.

Artificial intelligence (AI) is widely used in cases where a large amount of computation is required to obtain a specific result or where the human eye may miss important dependencies. The use of AI in IEEE 802.11 networks is particularly relevant , where attacks on network resources or clients can occur in a very short period of time. The use of artificial intelligence can effectively provide information about potential threats based on the analysis of previously collected data on the functioning of a computer network.

This can enable the timely detection of and response to abnormal activity and prevent possible cyber attacks, ensuring a high level of network security and reliability.

Today, there are a large number of services that allow remote management of TD and other network equipment from a cloud environment, and none of the above-described products that implement SP or SBB offer this level of solution. Unfortunately, there are no platforms that allow you to collect information from the air of Wi-Fi wireless networks and process it centrally to detect intrusions.

While configuring Wi-Fi TD does not require significant effort for a person with basic computer knowledge, deploying SP and SVV requires knowledge in the field of telecommunications and information security. Simplifying the deployment of such systems, centralised management of elements, and data processing that does not require end-user involvement can significantly increase the level of security not only for large corporate networks, but also for home and small office networks.

6 CONCEPT OF BUILDING AN INFORMATION SECURITY SYSTEM IN IEEE 802.11 WIRELESS NETWORKS USING INTRUSION DETECTION SYSTEMS AND DECOYS

6.2 DEVELOPMENT OF AN OPTIMAL LIFE CYCLE FOR DECOY SYSTEMS FOR IEEE 802.11 STANDARD NETWORKS

When using a decoy system, it is important to take into account its life cycle factors [157]. **Fig. 6.1** shows the cyclical sequence of measures for organising the functioning of a decoy system.

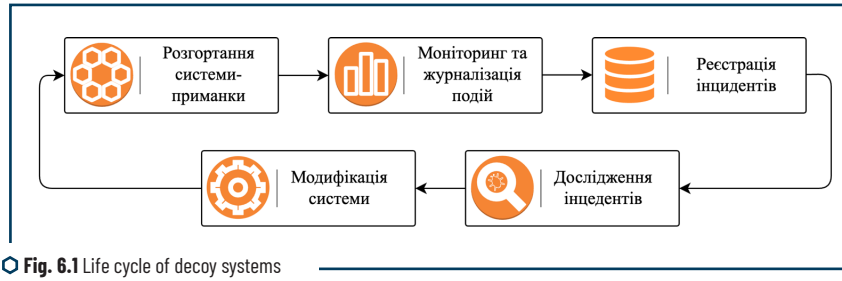

Fig. 6.1 Life cycle of decoy systems

The first stage is the deployment of the decoy system, during which configurations are applied in accordance with the needs of the system owner. The next stage is monitoring and logging of events, during which the performance of the decoy system is observed and all data on interaction with it is recorded. The third stage involves the registration of incidents. If incidents have been registered, the next stage is to investigate them. At the final stage, a decision is made on the effectiveness of the decoy system, and if the system has proven ineffective, its configuration is changed. More details on the life cycle of a decoy system will be provided later in this section.

Deployment. The first phase of SP configuration is the deployment of the resource with which the attacker will interact. It includes making decisions regarding:

– production systems that will be duplicated;
– the level of acceptable risk in the network;
– activities to be studied;
– network security policy;
– the level of interaction with the attacker.

The quality of an SP lies in its ability to lure and convince attackers of the legitimacy of the system they are attacking. A large number of vulnerabilities present in the network can be a marker that exposes the SP, while an insufficient number of vulnerabilities can reduce the chances of the SP being chosen by an attacker for interaction. Therefore, during the initial setup, it is necessary to add a number of vulnerabilities that are likely to appear in a real production network, which must be duplicated. At the same

time, well-known vulnerabilities that cannot exist in a production network ( ) due to, for example, the absence of certain software that is prone to such vulnerabilities, must be removed from the SP.

The most common attributes of this phase are:

1. Attack scenario:

— channel-level attack;

— IP-level attack (common for both wired and wireless environments).

2. Modification of IEEE 802.11 technology (a, b, g, n, ac, w).

3. Network security policy:

— open;

— use of one of the wireless security protocols (WEP / WPA / WPA2 / WPA3);

— enabling/disabling the quick network access function (WPS)

— application of client filtering mechanism by MAC addresses

— enable/disable broadcasting of the network identifier (SSID) on the air;

— use of additional authentication servers based on 802.1X/EAP protocols.

4. Basic infrastructure elements:

— wireless TD module;

— wired infrastructure module;

— client device module;

— wireless device module.

5. Level of interactivity:

— low interactivity (emulated);

— highly interactive (real infrastructure).

In addition, the network can provide different levels of interaction on each of the modules. Attributes considered in the SP modules:

1. Wireless TD module:

— emulated or real;

— number of generated access points (applicable to emulated).

2. Wired infrastructure module:

— emulated or real;

— presence or absence of services affected by the wireless network (TCP, UDP, ICMP, etc., as well as application-level services)

3. Client device module:

— emulated or real;

— type of traffic between the client and the TD;

— vulnerabilities available at the operating system level;

— vulnerabilities available at the software administration level;

— wireless network driver vulnerabilities.

4. Wireless device module:

— emulated or real administrative server.

The next phase is to monitor the activities of the attacker caught by the decoy, provided that they do not suspect that they are being watched. This phase includes logging inter-module interactions. Records of file changes, command entries, and services accessed must all be saved for further processing. In order to be able to log before, during, and after the attack, a traffic analyser is deployed alongside the bait. Software solutions such as KisMAC or Kismet, in combination with a packet analyser such as TCPDump or Wireshark, provide the user with an interface for passive monitoring of the wireless network. In radio frequency monitoring mode, traffic analysers can intercept frames without being connected to the TD. Data from the radio airwaves is recorded in files with the extension *.pcap and contains data about the attacker's illegal actions at the channel level. Depending on the needs, traffic analysers can be configured to listen to traffic on the same channel on which the TD SP operates, or to analyse traffic from other channels, which will allow you to assess the activity of wardrivers (people who search for and register Wi-Fi access points) in a given area.

The next phase is the storage of data on violations in the SP modules. This phase requires the ability to analyse the data from the previous phase in detail. All logged data (*.pcap files) and system log files are collected for analysis at a remote location via a wired or wireless network.

The most important stage, for which all the previous preparation was carried out, is the investigation phase. At this stage, the data collected over a period of time is analysed. In this phase, a detailed analysis of events that occurred on the air or within the SP is carried out using special tools or manually by analysts. In this phase, it is possible to find the location of the attacker at the time of the attack in relation to the wireless monitoring object and possibly find out about the tools (software, antennas, wireless card specifications) that were used during the attack.

Last but not least is the modification phase. This phase follows from the results of the previous phases. Modification of the infrastructure or architecture of the SP as a whole is carried out depending on information about attack vectors and newly discovered techniques.

The administrator may decide to increase or decrease the level of security by reconfiguring the TD module, for example, from the WPA wireless security protocol to WEP, WPA2 or WPA3.

More complex usernames and passwords can be implemented to prevent them from being guessed using a brute force attack, or, conversely, simple passwords can be added so that an attacker can gain access to a particular resource as quickly as possible. The same applies to TD software, which can be updated to the latest version to avoid exploitation of known vulnerabilities, or an old version can be used, which has well-known vulnerabilities.

During the modification stage, it is important to reset the TD settings to ensure that the configuration has not been modified during any of the attacks.

6.2 MONITORING AND LOGGING EVENTS

When all the methods by which an attacker could gain access to the network have been exhausted, it is likely that they will use a brute force attack. As is well known, a brute force attack is a method of solving mathematical problems that involves sequentially searching through certain data in order to find the

legitimate one. Thus, if this is a brute force attack that takes place online, the resource against which the brute force attack is being carried out will respond negatively until it receives legitimate data. Modern computer systems have the ability to set up preventive methods, such as a threshold for incorrect entries, which, once exceeded, requires the user to wait a certain amount of time before attempting to enter the data again. A similar method can be used to identify intrusions. For example, after k incorrect entries of certain data, it can be assumed that a brute force attack is being carried out.

Using the event log in IEEE 802.11 networks, the following types of attacks can be detected:

— a frontal attack on the server to obtain the real MAC address;

— a frontal attack on the server to obtain the hidden SSID of the network by a known MAC address;

— a brute force attack on administrative resources within the network.

The process of detecting all these attacks is similar. Let's take the last one as an example. There are cases when administrators or owners of access points do not change the default password for the administrative page of the AP or other resources that can provide additional privileges. In such cases, it is enough for an attacker to find the default login credentials on the Internet. Such a resource may be specially created by the network administrator and wait for a connection from a user who has previously scanned the network range. A connection to such a resource will serve as a marker of an attack [158].

In other cases, the attacker will select the password themselves or use a brute force attack. Usually, unsuccessful attempts by a user to access a specific page are logged in a special error file. By analysing such a file, a brute force attack can be identified using a set threshold for the number of incorrect user credential entries.

If an attacker has already gained access to the network, they will likely try to use this resource for their own purposes. Their goal may be to obtain user data, take control of the resource, or use it as a springboard for further attacks. Such attacks can be detected by monitoring the network and searching for anomalies.

Avenda System has offered a solution in its eTIPS product that allows you to detect MAC address spoofing. The method consists of continuously scanning devices with legitimate MAC addresses using the NMAP software package. NMAP allows you to scan devices on the network and, based on their responses — open TCP/UPD ports — identify the platform and services it provides. If a new device appears on the network and the administrator identifies it as legitimate, initial identification will take place using NMAP and, accordingly, a so-called fingerprint of the system will be obtained. During continuous scanning, the system will focus on detecting illegitimate client devices whose MAC addresses match legitimate ones but whose NMAP fingerprints do not match. There is a very low probability that an attacker will be able to replicate the configuration of a legitimate device. However, the use of such a product in wireless networks is quite ineffective, since wireless network clients mostly do not perform any server tasks that would allow obtaining a clear fingerprint of such a device. Therefore, it will be surprisingly difficult to detect the substitution of a client device in a wireless network using the NMAP software package.

In a man-in-the-middle (MITM) attack, an attacker can read and modify traffic between two nodes without them even knowing that the communication channel between them has been compromised. This attack is mainly carried out in Ethernet networks and is based on flaws in the ARP protocol. This involves creating a new entry point or updating an existing one in the victim's dynamic ARP table. After that, all traffic will pass through the attacker's workstation.

Detecting a MITM attack is possible because this type of attack generates a lot of noise. If the SVW is located in the middle of the network and traffic from other nodes does not pass through it, then it, like other nodes in the subnet, will receive a message with an ARP request from a node that should not have sent it. If the SBA operates on the main gateway of the subnet, it will receive an abnormally large amount of traffic from one node and no traffic from others.

Often, in order to attack a specific resource and not reveal their regular location, attackers use open Wi-Fi networks in public places. From the network to which the attacker has access, they can launch a denial of service (DoS) attack on any remote resource that may be accessible from that network. In addition to launching an attack from their workstation, attackers can exploit vulnerabilities in clients ( ) connected to the same network to amplify the attack. There are several types of DoS attacks, but the most common is one that aims to overload the resource's bandwidth.

Such an attack can be indicated by the detection of broadcast ICMP Echo Request packets, packets with no return path specified in the header, UDP Echo, Chargen or Disabled packets. Such an attack can also be identified by the set threshold of packets coming from the client. For example, a tool such as LOIC sends a large number of packets to the target at intervals of 0.00005 seconds.

Incident registration. The first step is to define what constitutes an incident. An incident can range from an attempt at unauthorised access at the system level to an attack on network traffic. To record incidents, you need to deploy an appropriate system that includes procedures, tools, and processes for recording incidents. This system can be a separate database, event logs, specialised software, and so on.

Incidents can be classified by type, importance, impact on the network, and other factors. This helps to determine priorities and levels of importance for different incidents.

Any incident must be documented in detail, including the date and time of detection, a description of the event, stored data, impact on the system, and other information that may be useful for further analysis. Information about incidents must be presented in the form of reports for network administrators and SPs. Events can be studied to improve network security strategies.

The purpose of the analysis is to understand and identify threats and attacks, as well as to develop strategies and measures to detect and prevent them. Analysis of attacker behaviour in Wi-Fi networks includes research into various methods and techniques used for intrusions, data theft, password cracking, and other types of cyber attacks.

In this context, researchers and network administrators need to study and implement various techniques and tools to detect and analyse abnormal events on the network, develop security strategies, and improve protective measures. Attacker behaviour analysis is an important part of modern cybersecurity and helps to improve the protection of Wi-Fi networks from potential threats.

The client's intentions can be understood as soon as they appear in the TD's coverage area. Service data at the channel level of the OSI model is open, broadcast over the air, and accessible to all IEEE 802.11 devices within the range of the source. In the same way that attackers intercept data about TDs (the frequency channels on which they operate, the methods used for protection, and the MAC addresses of connected client devices), data can be intercepted from the client devices from which the data originates. If the data is not typical for a normal client, the behaviour can be categorised as malicious.

Take, for example, an attack on a TD in which beacon transmission is disabled in order to hide the network identifier. If client devices are connected to the network, the attacker will be able to obtain data about the TD's SSID at the channel level by passively listening to network traffic, since clients will send packets containing information about the MAC address of the device to which the data is sent and its SSID. However, the attacker will not be able to obtain such data if no clients are connected to the TD with a hidden SSID. In this case, the attacker will be forced to perform a dictionary attack on the TD.

Another attack that can be detected on the air is an attack on the WEP wireless security protocol. Networks protected by the WEP wireless security protocol do not have an internal frame control mechanism. As a result, a frame can be replaced and transmitted to the network later. We can recognise this type of attack by checking the sequence numbers that are attached to each network frame. These sequence numbers typically help receiving devices sort frames in the correct order and detect missing frames. The sequence number helps to maintain the logic of operation, but for certain reasons, the session may be repeated after a while. Such a sharp change in order may indicate an active phase of the attack.

Today, the most robust security algorithm of the IEEE 802.11 standard is WPA3, but as mentioned above, an attacker can downgrade it to WPA2 thanks to a transition mechanism. And as we know, WPA2 is vulnerable to a number of attacks. One of the vulnerabilities is that an attacker can intercept a frame containing handshake information between the client and the access point. The attacker can passively scan the traffic on the air while the client is connecting to the access point, obtain this frame, and then decrypt it offline. Since this type of frame does not undergo authentication, an attacker can also generate such a frame on their workstation and send it over the air. Most tools that allow this attack to be carried out allow multiple such frames to be sent to achieve the goal. A large number of such frames can lead to denial of service. Therefore, the presence of a large number of deauthentication frames can serve as a marker for detecting both attacks on WPA/WPA2 wireless security protocols and denial of service attacks.

If WPA/WPA2 wireless security protocols are enabled, the WPS mechanism can also work with them. As already mentioned, an attack on WPS is based on a brute force attack. This attack leaves a clear trace, as a large number of false numerical sequences that do not match the factory PIN code are sent to the TD in a short period of time [160].

All of these types of attacks can be identified in the air before the attacker successfully completes one of them. Thus, this method allows you to respond to an attack in advance and gain precious seconds to divert attacks to non-production resources.

System modification. Conducting an analysis of the effectiveness of the SP is the last but no less important stage in the process of its use and support. This analysis helps to assess how well the SP meets its goals and objectives, as well as to identify its shortcomings and opportunities for improvement.

It is important to determine whether the SP's performance meets the organisation's security objectives. Has the system been able to detect and record unwanted activity? Have potential threats been identified? Performance analysis helps to identify weaknesses and vulnerabilities in the SP that could be exploited by attackers to bypass security.

Another important aspect is assessing how efficiently resources are being used, such as disk space and computing resources.

It is important to determine how many false positives and false negatives the SI generates. False positives can lead to unnecessary work for staff, while false negatives can miss a potential threat. It is important to assess how the SI interacts with other components of the security infrastructure, such as IPS, antivirus programmes, etc.

Based on the results of the analysis, strategies can be developed to improve network security and the SI. The final stage of the effectiveness analysis helps to understand how the SI functions in real-world conditions and contributes to its further improvement and adaptation to new threats.

6.3 RESEARCHING POSSIBLE UNMASKING SIGNS IN DECOYS

As is well known, cybercriminals and information security specialists are in a constant race. After the invention of honeypot technology, this race took on a new form, as the victims are now the attackers.

Although SPs are designed to gather evidence against attackers and reduce the likelihood of attacks on production systems, as mentioned above, SPs themselves, due to their imperfections, cause damage rather than benefit. In the best case scenario, lures can be identified by attackers and abandoned. In the worst case, SPs can become a springboard for further attacks on AS.

In this section, we will look at possible telltale signs for decoy systems that can be used in wireless networks.

Detecting a decoy at the channel level of the OSI model . If a corporate AP is configured to operate without using vulnerable security mechanisms, this may indicate a lack of security in such a network or the possible use of a decoy.

A tool such as FakeAP helps to deploy one or more access points. If this tool is used to deploy a single TD, it will not differ from any other in the air, and using it in this way is not advisable. FakeAP is mostly used to deploy a large number of access points. This approach allows you to distract attackers from real production networks.

To deploy an AP using FakeAP, you need to specify the following data:

— the channel on the air on which the AP will be deployed;

— the name of the AP that will be displayed on the air.

Also, an AP can be cloned from an existing one. To do this, you need to specify the name of the AP that you want to clone. After that, the AP will be as similar as possible to the real one.

A large number of access points with the same name but different MAC addresses may indicate that a specific coverage distribution mechanism is being used, such as Wireless Distribution System (WDS) or Mesh, or that SP is being used. The use of SP, which generates multiple fake access points using a single physical device in IEEE 802.11 wireless networks, can be distinguished from coverage distribution mechanisms by measuring the power from all access points with the same name. The power of access points operating in coverage distribution mode will show different values because they are physically separated from each other. The power from access points emulated by the FakeAP software package will be approximately the same. This technique can be used by attackers to identify that there is a fake access point on the air.

Detecting a decoy at the network layer of the OSI model . Once an attacker is inside the network, it is likely that they will continue their attack to gain access to other resources within the network. The first step towards achieving this goal will be to scan the network. Once the attacker has obtained a list of network nodes and the services running on them, they can proceed with the attack. In this case, the attacker's actions will be logged.

Today, there are already signatures that can report interaction with SP. If the attacker runs a scanner for known lures and they are identified, then such a system will likely be abandoned immediately [161].

If the SP is a service that is frequently visited in a real environment, then requests to it must be present, otherwise it may become a demasking factor. Often, in order to create the illusion of SP legitimacy, various software tools are launched that generate network traffic.

In network security, there is a method for detecting attacks by analysing traffic and searching for anomalies in it. This method is used by attackers to detect SPs in the networks they attack. The main problem is that software tools that generate traffic do so monotonously, without variation, which is not characteristic of a legitimate user. Therefore, in the case of attackers, one of the signatures of SP can be the identification of monotonous traffic [162].

Detection of a lure at the application level of the OSI model . Being inside the network, it is only natural that an attacker will want to gain administrative access to a specific network link. In the case of an SP for a wireless network, at least one network node must have the ability to administer the wireless network.

As is well known, each network equipment manufacturer has a specific range of MAC addresses that can be assigned to its equipment. Also, for the most part, each manufacturer of Wi-Fi access points develops its own control panels.

Access points are ready to use right out of the box, but with default settings. Each manufacturer has its own template for assigning default SSIDs to access points.

Therefore, the administrative resource software, the SSID of the access point, and its MAC address must match, as any discrepancy may arouse suspicion in an attacker [163–166].

6.4 DEVELOPMENT OF A CONCEPTUAL MODEL OF AN INFORMATION SECURITY SYSTEM USING INTRUSION DETECTION SYSTEMS AND DECOY SYSTEMS

The use of cloud computing for organisations has many advantages that help improve the efficiency, security and adaptability of their information systems and business processes. By using cloud computing, organisations can avoid significant costs for purchasing, installing and maintaining their own servers and equipment. They can use the ready-to-use infrastructure of cloud providers. Cloud providers allow you to instantly increase or decrease the amount of resources depending on the needs of the organisation. This allows you to maintain optimal performance and reduce costs when resources are not needed.

Cloud computing allows you to use high-performance computing resources and hardware without having to wait for equipment to be purchased and deployed. Many cloud providers have extensive data networks and backup measures, making their infrastructure reliable and resilient [167].

Cloud computing providers typically take on the responsibility of updating and maintaining the infrastructure, as well as ensuring security, so that the organisation can focus on its core tasks.

One of the biggest advantages of cloud computing is the cost savings on IT infrastructure. These technologies allow organisations to pay only for the resources they use, resulting in lower costs compared to traditional infrastructure solutions.

The concept of Anything as a Service (XaaS), which is also synonymous with cloud computing, represents any Internet technology as a service. The most well-known examples of XaaS are Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) models.

There is also a model for providing cybersecurity services called Security as a Service (SECaaS). Using the SECaaS model, organisations or individuals can access various security services via the Internet. Instead of installing and maintaining various products and solutions to protect their infrastructure and data themselves, customers can use cybersecurity services that are provided remotely and maintained by third-party providers.

SECaaS is a model in which a service provider with large computing power that provides information security services integrates its services into the customer's corporate infrastructure. This approach is often more cost-effective than having to deploy a security system on your own. In this model, the customer does not need to worry about maintaining the security system, as all work to ensure the security system's operability is the responsibility of the service provider. Such services often include authentication services, antivirus software, anti-spyware, SVA, and security event management [168, 169].

This section proposes a new sub-concept of SECaaS, tentatively named "Wireless Honeypot as a Service" (WHaaS). WHaaS is a service aimed at ensuring information security in IEEE 802.11 standard frequency range networks by deploying fake access points and monitoring them continuously. A schematic representation of the WHaaS model is shown in **Fig. 6.2**.

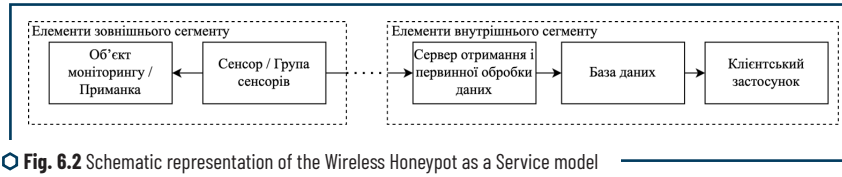

Fig. 6.2 Schematic representation of the Wireless Honeypot as a Service model

Elements of the Wireless Honeypot as a Service model . IEEE 802.11 standard TDs, which are aimed at the small or home office segment, usually do not have the ability to perform any role other than providing access to a specific network segment. Installing any additional software is a very difficult task due to architectural incompatibility or insufficient computing resources. Any workstation with an integrated or discrete IEEE 802.11 network card can be a Wi-Fi sensor. Single-board computers are convenient for this purpose. This type of workstation has sufficient resources to run operating systems such as Windows or Linux. Also, microcontrollers that already have integrated Wi-Fi modules can handle certain types of tasks related to IEEE 802.11 networks.

Any two network nodes can communicate with each other provided that they are located within the same private network or are located in different networks but have been assigned public IP addresses. However, assigning public addresses to elements involved in the collection and processing of service information is a costly solution, as there may be quite a few such elements. In addition, external access to network nodes is dangerous because they can be easily attacked from the Internet.

Wi-Fi networks that do not have distributed coverage usually cannot cover a large area. The ability to connect a device to a network is highly dependent on antenna gain, and in the case of energy-efficient devices, their range will be up to 50 metres in the absence of interference. In addition, to improve security, it may be necessary to cover an area larger than the controlled zone. In this case, the sensors may be located at a distance that does not allow them to communicate with each other. Therefore, in order to expand the coverage radius, in the WHaaS model, it is advisable to create the possibility of being physically connected to different networks, since sometimes direct communication between sensors may be impossible.

In order to ensure communication between sensors that may be connected to different networks, they must belong to a common resource. Such a resource can be created using virtual private networks (VPNs). After connecting to the VPN server, the sensors will have access to all elements of the network and vice versa.

Once the external elements of the system are accessible on the network, communication with them can be implemented using a specific network communication protocol. Depending on the operating system, this can be RDP for Windows and SSH for Unix. The element that communicates and controls remote elements will hereinafter be referred to as the command centre (CC). The CC will allow you to install and update software or execute specific commands on remote elements of the complex. Scenarios can be generated according to the operating system installed on a particular external element [170].

The C&C is an important component of the information security infrastructure based on the WHaaS model. One of the key aspects of the C&C is the database (DB), which plays an important role in the functioning of this system. The DB is used to store the history of activity on the network and the ether. The SVV receives data about network traffic, user interactions, and malicious activity. This information is recorded in the DB and stored for further analysis. This allows the system to track changes in network activity and recognise anomalies.

An important task of the database is that it allows the SVB to build intelligent models of attacks and anomalous activity based on historical data. It learns to recognise typical intrusion scenarios and respond to them. They are also used to store information for creating reports and analytics on network security. This includes identifying persistent trends or security incidents, which helps network administrators make informed decisions about improving security or modifying the SP.

Once the data has been collected, it must be analysed. Data analysis is an important component of any information security system. The main task of this component is to identify deviations from the norm, detect anomalies based on historical data, recognise potentially dangerous situations based on signatures, etc. Thanks to data analysis, the protection system can respond to detected threats in a timely manner, taking measures to prevent or eliminate them. In addition, data analysis helps to improve intrusion detection methods and algorithms, making the system more effective in detecting new and unknown threats.

An important component of any modern system is a resource where the user or administrator can monitor the status of the system, configure it, and receive notifications about certain events. This functionality

is usually provided by client applications deployed on a specific server. An application server is an environment with a software application that receives data from a database and sends it to the user interface, where the user can work with the aggregated data.

All components of the internal segment must ensure uninterrupted service provision, as the unavailability of the VPN server or database puts the operability of the entire system at risk. The best fault tolerance and survivability of the computing infrastructure can be achieved through the use of cloud computing. Cloud computing has many advantages, such as scalability, flexibility, accessibility, cost reduction, etc. [171–173].

Therefore, based on the above, we can define a minimum set of elements for implementing an information security system based on the WHaaS model:

1. External segment:
— Wi-Fi sensors.
2. Internal segment:
— VPN server;
— command centre server;
— database;
— data analysis server;
— application server.

Communication between elements of the external segment of the WHaaS model . Using a VPN server enables communication between the cloud infrastructure, which must be located in a private network segment, and sensors that do not have public addresses.

The VPN server must be assigned a public address, as it must be accessible from anywhere in the world or from specific networks. Depending on the selected VPN server, certain TCP/IP ports must be open and all others closed. The power of the VPN server will be determined by the number of elements in the external segment.

As soon as any element is connected to the VPN server, all available resources of the private subnetwork of the cloud computing network will immediately become available to it.

Once the external elements become available in the private subnet of the cloud computing network, the sensors will be able to read and write data to the database. The sensors will continuously collect data about events in the IEEE 802.11 frequency range and write them to the database.

To access the database server, you need to connect to the network using a VPN server. Depending on the type of database selected, the workstation on which the database is deployed must have specific TCP/IP ports open for communication with it and all others closed. The power of the database server is determined by the amount of data coming in for recording.

Each software product has its own life cycle, both for its own software and for software downloaded from external repositories. Most attacks target systems with outdated software. The relevance of software is monitored through a command centre, where both regular and irregular commands from users are generated.

For secure interaction, only remote connections via a private network within a VPN are allowed on external elements. This involves opening only specific TCP/IP ports for communication with the service that provides the remote connection, while all other TCP/IP ports must be closed. For example, for the Linux operating system, such a service is Secure Shell (SSH), which uses port 22/TCP.

When a user accesses a web application, the application server begins to connect to the database. Data is selected from the database and, after processing, sent to the user.

When a user decides to execute an irregular command on the sensors, they must first log into the user web interface and form their requests. After confirmation, the change data is sent to the command centre.

Depending on the chosen communication platform, certain TCP/IP ports must be open on the workstation acting as the command centre for communication with it, and all others must be closed.

Any cloud service must have a user interface for presenting and managing information. To access the collected and processed data, the user utilises a client interface that selects data from the database and displays its aggregation. Based on constantly updated data from the database, which is fed by sensors, the data processing server determines the presence or absence of an intruder in the monitored airspace.

Presenting the most important data on a single page has become a trend among modern cloud technology providers. For the WHaaS model, such data may include:

– number of identified IEEE 802.11 devices;
– signal strength from controlled access points;
– detected attacks;
– results of external component performance checks.

The concept of such a page is presented in **Fig. 6.3**.

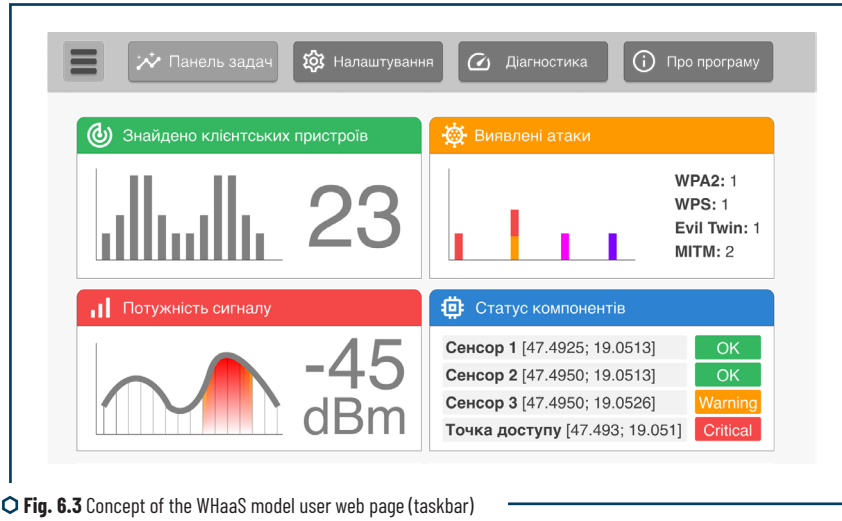

Fig. 6.3 Concept of the WHaaS model user web page (taskbar)

Even if external elements have an auto-configuration function, the user must be able to set the configuration at their discretion. In addition, detailed monitoring must be available (**Fig. 6.4**).

In order to apply the settings manually, the user must connect to the user web page and set the required configurations. After the user clicks the button to apply the changes, the data will be sent to the

command centre, where a script for the external element will be generated, after which the script will be sent and executed on the external element (**Fig. 6.5**).

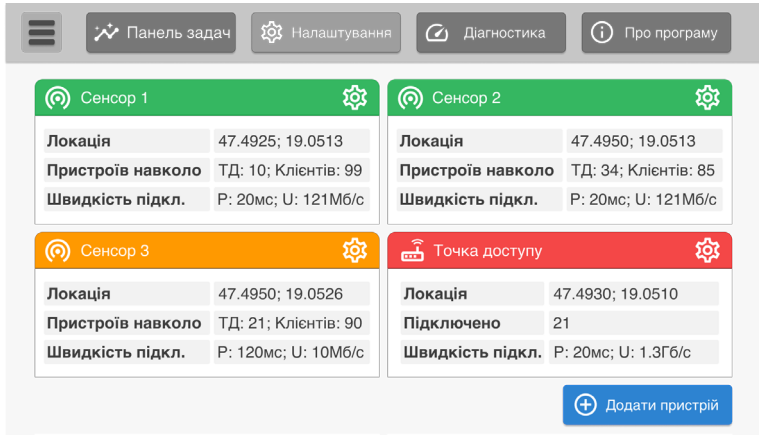

Fig. 6.4 Concept of the WHaaS model user web page (settings panel)

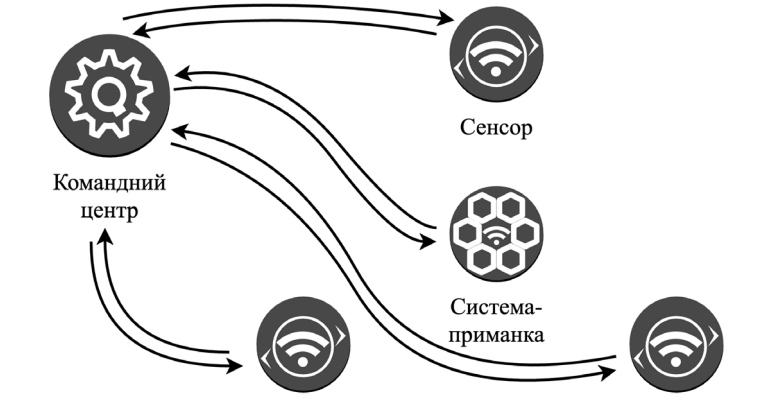

Fig. 6.5 The process of applying settings to external elements

In general, from all of the above, we can present the overall architecture of the system and the communication between its elements (**Fig. 6.6**).

As can be seen in **Fig. 6.6**, there are several possible communication flows through which data circulates within the system. The first is the sensor administration process. For this, both the administrator and the sensors must be connected to the VPN server, which will enable communication with the command centre.

The second is the process of recording data by the sensor in the database. After establishing a connection with the VPN server, the sensor gains access to the database, where it can record raw data. This brings us to the third communication flow — the process of data processing and analysis. At this stage, raw data is processed and prepared for display on the client application.

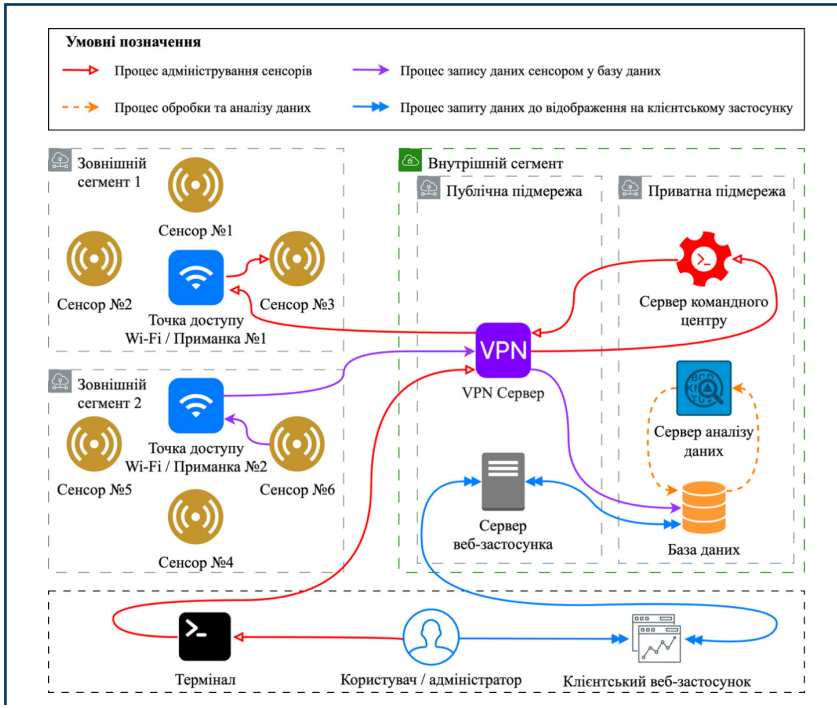

Fig. 6.6 Basic elements and communication of elements of the IEEE 802.11 network security system based on the WHaaS model

The fourth is the process of requesting data for display on the client application. At this stage, the user connects to the web application server, which is located in the public subnet of the internal segment. Since the web application server is located in a public subnet, access to it is unrestricted.

7 DEVELOPMENT AND OPTIMISATION OF THE INFORMATION SECURITY MODEL FOR IEEE 802.11 WIRELESS NETWORKS.11 USING INTRUSION DETECTION SYSTEMS AND DECOY SYSTEMS

7.1 DEVELOPMENT OF A METHODOLOGY FOR TRACKING ATTACKERS

Even after an attack has been detected, bringing the attacker to justice is not an easy task, as the attack may be carried out outside the controlled area. In the case of an attack on WPA/WPA2 wireless security protocols, the attacker only needs to intercept one packet, which takes less than a minute. However, the features of the IEEE 802.11 standard that allow attackers to carry out attacks on them can help in finding the device that participated in the attack, which in turn can lead to the attacker [174].

Features of search packets in the IEEE 802.11 standard as a mechanism for reconnecting to the network. If a user device has ever been connected to any IEEE 802.11 wireless network, the connection data has already been stored on the device. An exception may be a situation where the user has deleted the network information from the device or has set a parameter in the operating system settings that prohibits the storage of Wi-Fi wireless network settings.

This feature of the IEEE 802.11 standard allows clients to easily continue working with the AP even after the user has left its coverage area. This works because when client devices are not connected to any network, a mode is activated that searches for known access points. At the packet level, this is organised in such a way that client devices send search packets (Probe Request) to the network, and if the AP is within the device's range, it uses the authorisation data already available on the device (**Fig. 7.1**).

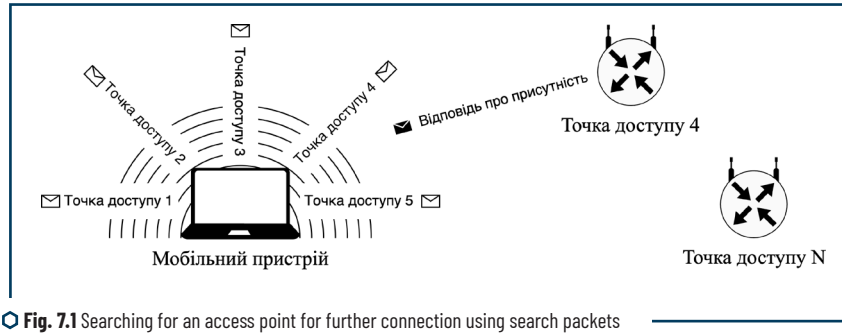

Fig. 7.1 Searching for an access point for further connection using search packets

If the AP responds with a Probe Response packet on the air, an authentication process will be attempted (**Fig. 7.2**).

To see probe packets on the air, you do not need special equipment – you just need to activate the monitoring mode of your Wi-Fi network card. This feature is often used by attackers to find the network

identifier if it is hidden. The client device sends a certain number of search packets into the air, after which the attacker intercepts them and sorts through the identifiers when attempting to connect to the network.

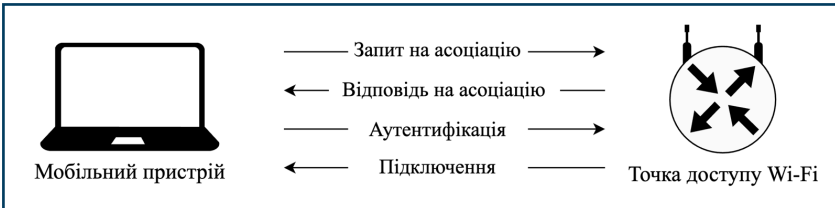

Fig. 7.2 Authentication process for network access after the access point responds with a presence packet on the air

It should be noted that attackers are also users of wireless networks and, as a rule, even an average Internet user may have several devices that work via Wi-Fi. These can be smartphones, smart watches, and other IoT devices. When attackers enter their target's coverage area, their devices will send search packets over the air. In the best case scenario, it would be ideal if the device from which the attack is being carried out was already connected to a Wi-Fi wireless network. In this case, in addition to detecting the attack and linking it to a device with a specific MAC address, we will also obtain data about which networks the attacker has connected to previously. Of course, the attacker can use a so-called disposable operating system that is downloaded from a flash drive and resets all settings after rebooting. However, as mentioned above, the attack device is unlikely to be the attacker's only device.

Thus, collecting search packets from other devices identified in the same monitoring sector can provide more information about the attacker (**Fig. 7.3**).

The WHaaS model proposed above suggests the use of independent sensors to detect intrusions. Through a specific communication channel, the sensor transmits service data, which includes information about detected attacks, devices identified alongside the device from which the attack was carried out, and networks to which these devices were previously connected [175–177].

Detection of previous locations of the attacker . Wardriving is an activity in which a person travels in a car or other vehicle, usually with a computer and specialised equipment, to search for and identify Wi-Fi wireless networks. Wardriving combines two English words: "ward" (walking around, travelling) and "driving", as the main idea is to drive around an area looking for active Wi-Fi networks.

The main goal of the above activity is to find and record information about available Wi-Fi networks, including their identifiers (SSID), signal strength, encryption, geolocation, and other parameters. Today, there are resources where researchers share such data. One such service is the Wireless Geographic Logging Engine (WiGLE).

WiGLE is a resource for sharing information about Wi-Fi access points from around the world. Users can register on this web resource and upload data such as GPS coordinates, SSID, MAC address, security type, and other metadata about the access points they have found in order to share this information with other users.

As mentioned above, intercepted search packets from the attacker's devices may contain data about the access points to which they were previously connected. Once an attack is detected, this data can be sent to the WHaaS system's data processing server. The names of the networks to which the attacker's devices were previously connected are transmitted to the wigle.net service, which returns an array with data about the location of the networks. However, along with the necessary data, we receive a large amount of redundant information, such as the channel on which the TD operates, and other information that may be confusing. After filtering, this data must be sorted by location. This will allow us to obtain an array of data from each location where the TD was found with the parameters collected from the attacker's devices.

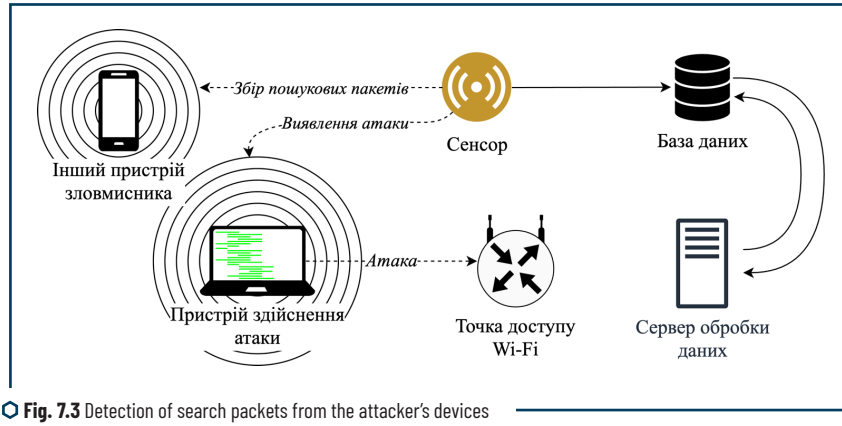

Fig. 7.3 Detection of search packets from the attacker's devices

It is quite possible that the attacker will come from another city or even another country. Nevertheless, the sector in which the attacked TD is located will obviously have a higher priority, even if a larger number of access points are identified in another territorial unit. It is important to take this factor into account, as the protection system may be located in a city that is smaller than, for example, the country's capital.

Once you have a set of coordinates, you can roughly estimate the distances between them, starting with the current territorial unit.

To find out the approximate distance between points, you need to use the formula for calculating Cartesian distance on a plane (7.1)

$$d=\sqrt{((\varphi\_2-\varphi\_1)^2+(\lambda\_2-\lambda\_1)^2)}, \quad (7.1)$$

where λ – longitude (degrees), φ – latitude (degrees).

Once the distances between all points in the territorial unit have been found, they must be added together to determine the density of access points (7.2).

$$D=\sum d\_i=(d\_1+d\_2+...+d\_n). \quad (7.2)$$

According to formula (7.2) D→ 0, because the closer the *TDs* are to each other, the smaller the value of *D* will be. Accordingly, the smaller the value of D, the greater the probability that the territorial unit in which the offender resides has been determined. The priority is indicated in (7.3).

$$Pr=[D\_min,...,D\_max\ ]. \tag{7.3}$$

Client devices often retain data about *TDs* that clients have only attempted to connect to by randomly guessing the password. Let's describe the situation: a client tries to access the Internet and starts searching for all possible access points nearby on their device. Several of them prohibit further interaction due to, for example, unsuccessful authorisation. In this case, the data with which the client attempted to authorise has already been saved on their device, and if, in the future, these access points are identified by sensors, the locations where they are located can be given a high probability coefficient of the attacker's previous presence.

With data on the geographical location of access points that were likely visited by the attacker, it is possible to create a graph with weights (the number of access points found) and the distance to the SP (**Fig. 7.4**).

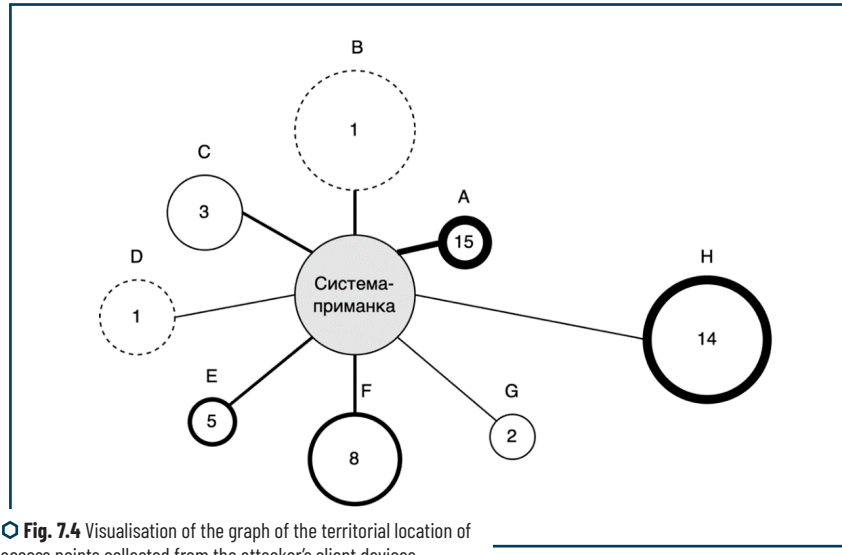

Fig. 7.4 Visualisation of the graph of the territorial location of access points collected from the attacker's client devices

Additional visualisation of the graph will help you understand which territorial units to focus on. The thickness of the connection line is the strength of the connection, which is understood as the distance from the AP to the territorial unit and is calculated using the same formula (7.1). The radius of the circle is the

size of the territorial unit. The thickness of the circle's outline, like the number in the middle, indicates the number of access points found with names collected from the attacker's client device.

According to **Fig. 7.4**, although nodes E, F, and H are quite likely locations for the attacker to be, it is still worth focusing on territorial unit A, as it is closest to the AP, is a small territorial unit, and has the largest number of access points found among the others. As mentioned above, there is a high concentration of Wi-Fi access points in large cities, and it is quite likely that all the access points identified on the attacker's device will be found in a large city. It is also important to understand that the Wi-Fi network identifier is not unique, so false positives are likely, especially in large territorial units.

In addition to the distance from the SP, you need to consider the uniqueness of the names of access points scanned from the attacker's device. For example, 14 access points with the specified search name were found in location H, 10 of which are unique, and in location A, 15 were found, 5 of which are unique. However, there is another parameter that can provide more information than all of the above. This parameter is a single access point or group of access points with unique network name identifiers that were not found in any territorial unit (**Fig. 7.5**).

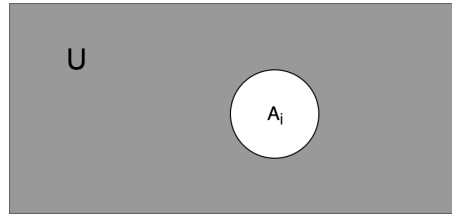

Fig. 7.5. Unique element in a set of values

The set of all access point names from all locations is a universal set U = A∪ B∪ C∪ D∪ E∪ F∪ G∪ H. A set containing all elements of a universal set except for element $A_{(i)}$, which in this case is a unique element, is called the absolute complement of $A_{(i)}$ and is denoted as or $A^{(C)}$ or $C_{(A)}$. Formally: .

7.2 DIAGNOSTIC MODEL OF A WIRELESS NETWORK DECOY SYSTEM FOR THE IEEE 802.11 STANDARD

The basis for developing criteria for a diagnostic model of an IEEE 802.11 wireless network SP is based on an assessment of the complexity of circumventing their protection methods. As already mentioned in previous sections, each of the protection methods can be compromised depending on the skills and technical equipment of the attacker.

A TD, or in our case a SP, that has insufficient or no protection can easily become a target for individuals who want to take advantage of free Internet access or arouse the suspicion of experienced attackers.

General diagnostics of a decoy system for an IEEE 802.11 network . As mentioned above, WEP is the weakest of the wireless security protocols, so its use by a TD will likely be perceived as a mistake on the part of the network administrator or the presence of outdated equipment on the network.

A certain combination of factors can significantly reduce the likelihood of an attacker interacting with the SP. For example, in most TD configurations, a connected client is required, without which an attack becomes impossible. By collecting metadata from the 2.401–2.483 GHz frequency range using a network card operating in monitoring mode, an attacker can obtain complete or partial data about the AP and its clients.

The vulnerability mentioned above in the WPA/WPA2 wireless security protocols can only be exploited if a client is connected. The exception is a configuration with the WPS mechanism enabled, which does not require a connected client for an attack, since the PIN code is only brute-forced against the AP.

To attack WPA/WPA2 wireless security protocols, an attacker needs to obtain a special handshake packet between the AP and a client that knows the network key. This packet is intercepted by the attacker when the client connects to the network.

In configurations where mechanisms such as hidden SSID, MAC address filtering, or WPA2 wireless security protocol with WPS disabled are present and there are no connected clients, the situation becomes more complicated. In such cases, the attacker will have to wait for a client to appear or launch a brute force attack to obtain the SSID or MAC address. As for MAC addresses, they are more likely to be obtained through brute force, since the number of possible MAC addresses is $16^{(6)}$, while for SSIDs, the possible options are limited only by the length of the string from 1 to 32 characters, which can be selected from the UTF-8 table.

To configure a SP using WPA/WPA2 wireless security protocols, a key must be set, which can be found during brute force attacks using publicly available dictionaries. Manufacturers of modern routers implement a timeout mechanism after a certain number of failed attempts to counter brute force attacks on the WPS mechanism. This functionality can prevent an attacker from accessing the SP.

Based on the above data, we will formulate a truth table of combinations of IEEE 802.11 wireless network security mechanisms (**Table 7.1**) and construct a block diagram of the algorithm for testing the SP for vulnerability to attack (**Fig. 7.5**).

Table 7.1 Possible combinations of IEEE 802.11 wireless network security mechanisms

No	Open	WEP	WPA	MAC filter	Hidden SSID	WPS	Client device presence
1	2	3	4	5	6	7	8
1	+	-	-	-	-	-	Not necessary
2	+	-	-	+	-	-	Required
3	+	-	-	-	+	-	Required
4	+	-	-	+	+	-	Required
5	-	+	-	-	-	-	Not necessary

Continuation of Table 7.1

1	2	3	4	5	6	7	8
6	-	+	-	+	-	-	Required
7	-	+	-	-	+	-	Required
8	-	+	-	+	+	-	Required
9	-	-	+	-	-	-	Required
10	-	-	+	+	-	-	Required
11	-	-	+	-	+	-	Required
12	-	-	+	+	+	-	Required
13	-	-	+	-	-	+	Not necessary
14	-	-	+	+	-	+	Required
15	-	-	+	-	+	+	Required
16	-	-	+	+	+	+	Required

Fig. 7.5 does not take into account the WPA3 network protection mechanism, since there is a transition mechanism for devices that are not capable of supporting it, and therefore it can be downgraded to WPA2.

To assess the complexity of overcoming the conditional protection of the SP, we will evaluate it using the coefficients of the hierarchy analysis method, based on criteria such as the time required to overcome the protection and the availability of the appropriate hardware that the attacker will have to use (**Table 7.2**).

Table 7.2 Assessment of the complexity of overcoming the conditional protection of the decoy system

Estimation criterion method of protection	Time to overcome	Availability of appropriate hardware	Difficulties during the attack
1	2	3	4
WEP	≈ 30 min	A basic Wi-Fi network card is sufficient to carry out this attack	If the client is not connected to the TD, the attacker needs to perform a spoofed authentication attack
WPA/WPA2	5 min > t > ∞	To carry out this attack, you need to have a network card with which you can send a deauthentication packet	If the client is not connected to the network, , the attack cannot be carried out because it is not possible to intercept the handshake packet. The attack becomes more difficult as the number of characters in the key and the number of

Continuation of Table 7.2

1	2	3	4
WPA3	10min > t > ∞	The same applies as for attacks on WPA/WPA2. In addition, the transition mechanism for WPA2 devices must be enabled (WPA2/WPA3)	The same applies as for attacks on WPA/WPA2. In some cases, access to the device from which the connection to the network was made is required
MAC Filtering	1 min > t > ∞	To carry out this attack, all you need is a basic Wi-Fi network card	The attack is complicated if there are no clients connected to the network, in which case a brute force attack is required
Hidden SSID	1 min > t > ∞	To carry out this attack, all you need is a basic Wi-Fi network card	The attack is more difficult if there are no clients connected to the network
WPS	1 min > t > ∞	A basic Wi-Fi network card is sufficient to carry out this attack	The attack is more difficult or impossible if the WPS version is > 1

This method allows you to obtain the ratio of scale weights by pairwise comparisons with a small deviation [178, 179]. Actual measurements or subjective opinion are used as coefficients. The output is the ratio of weights and the consistency index.

In the standard version of the hierarchy analysis method, any group of characteristics is evaluated using a scale of coefficients from 1 to 9 (**Table 7.3**). This table is used to form a weight matrix for each of the elements in comparison with each other (3.4).

N=|(1&a_12&…&a_1n@a_12^(-1)&1&…&a_2n@…&…&…&…@a_1n^(-1)&a_2n^(-1)&.&1)| (7.4)

Table 7.3 Scale for evaluating coefficients in the hierarchy analysis method

Numerical value	Definition
1	Equal value
3	No significant advantage of one value over another
5	Significant advantage of one value over another
7	Visible advantage of one value over another
9	Absolute advantage of one value over another
2, 4, 6	Average judgement between two adjacent judgements

Based on **Tables 7.2 and 7.3**, we will assess the complexity of bypassing protection mechanisms in relation to each other (**Table 7.4**).

Table 7.4 Comparison of the complexity of bypassing Wi-Fi access point protection mechanisms

Protection method	Advantage/disadvantage ratio	Method for comparison
MAC filtering	½	Hidden SSID
	5	WEP
	8	WPA/WPA2
	9	WPA3
	7	WPS
Hidden SSID	2	MAC filtering
	5	WEP
	8	WPA/WPA2
	9	WPA3
	7	WPS
WEP	1/5	Hidden SSID
	1/5	MAC filtering
	6	WPA/WPA2
	9	WPA3
	5	WPS
WPA/WPA2	1/8	Hidden SSID
	1/8	MAC filtering
	1/6	WEP
	4	WPA3
	1/4	WPS
WPA3	1/9	Hidden SSID
	1/9	MAC filtering
	1/9	WEP
	1/4	WPA/WPA2
	1/8	WPS
WPS	1/7	Hidden SSID
	1/7	MAC Filtering
	1/6	WEP
	4	WPA/WPA2
	8	WPA3

To further normalise the matrix, it is necessary to find the sums of the coefficients for each of the columns N (7.5):

S_i=∑_(i=1)^n..a_i=a_i1+a_i2+...+a_in] (7.5)

Using the sums of the column coefficients, we can derive the normalised matrix (7.6)

S_i=∑_(i=1)^na_i=a_i1+a_i2+...+a_in] (7.6)

By summing the coefficients of each row of the normalised matrix and dividing the sum by the number of coefficients in the row according to formula (7.7), we can obtain the weight of each of the protection mechanisms.

x=[(∑row_1/n@∑row_2/n@...@∑row_k/n)] (7.7)

Thus, this mathematical apparatus allows us to evaluate and compare each of the protection mechanisms with each other. If new protection mechanisms appear, the table will be adjusted and the protection mechanisms will be re-evaluated by an expert. Today, the WPA3 protection mechanism is the most effective, but WPA2 remains the most widespread, so it is necessary to focus on detailing the assessment of this protection mechanism.

7.3 DETAILED ASSESSMENT OF THE COMPLEXITY OF BYPASSING THE CONDITIONAL PROTECTION OF A DECOY SYSTEM WITH WPA/WPA2 WIRELESS SECURITY PROTOCOLS

The expected result of this study is to obtain a scale of complexity P_k of the WPA/WPA2 key for the SP of the IEEE 802.11 wireless network standard for the end user based on the criteria a_{ij} of the *N* matrix obtained by the hierarchy analysis method. The result of the correct configuration of the SP is the test data *t(BF)*, which predicts the time required to crack the key using a brute force attack. The test is performed based on data from dictionary *D*, namely the current dictionary *d(i)*,the result of which is a set *l(i)—l(i-1)* of unique keys *l={k(1), k(2),..., k(n)}*, and the key search speed *S*, which is measured from a specific reference system. By comparing this method with full virtualisation and operating system-level virtualisation methods, an environment for conducting a distributed brute force attack will be selected, which will allow finding the optimal method for assessing the conditional security of the SP [180].

There are two modifications of this method: WPA2 Personal and WPA2 Enterprise. The main difference between them is in the management of AES encryption keys. In the case of WPA2 Personal, keys are stored on each individual user device, which is typically used in home or small office (SOHO) networks. The key in the WPA2 Personal mechanism is the same for all users. For corporate applications, a dynamic key is used, which is individual for each user. A special server, usually RADIUS, is responsible for generating login-password pairs in WPA2 Enterprise.

The WPA2-PSK (Pre-Shared Key) protocol allows mobile devices to exchange data with the TD using the AES encryption method. In cryptography, PSK is a key that is pre-identified between two nodes using a specific secure channel before it is used. PSK is obtained from the key using the PBKDF2 password-based key generation standard with the SHA1 hashing algorithm as a pseudo-random function. PSK is 32 bytes (256 bits) long and is often represented as 64 hexadecimal characters.

The PBKDF2 standard is in turn described by the PKCS#5 standard. In this case, the password must be between 8 and 63 characters long. Characters are converted to machine code using the ASCII encoding table, so only these characters can be used in passwords.

In PBKDF2, the binary password is used as the key to the HMAC function. The TD SSID identifier is used as random additional information, known as "salt". The salt with the counter value is used for the initial input to the HMAC function. After that, the previous output is used as input for subsequent iterations until 4096 HMAC calculations are reached. Note that although the HMAC algorithm can generate output of arbitrary length, in your case a fixed length format of 256 bits is used. In comparison, the SHA1 hash function returns output of only 160 bits.

Understanding that the output of the PBKDF2 function is used as a Pre-Shared Key (PSK), the PSK can be defined as a Pairwise Master Key (PMK) directly in the four-way handshake process (7.8):

$$PSK = PBKDF2(HMAC - SHA1 \mid Passphrase \mid SSID \mid 4096 \mid 256). \quad (7.8)$$

Passwords used by ordinary consumers of modern Wi-Fi equipment are usually not complex. They can be a combination of numbers from one to nine, letters on the keyboard that are located next to each other, or simple combinations such as a date of birth. Today, there are a large number of generated dictionaries with simple phrases used by ordinary users. These dictionaries are available on the Internet or in specialised operating systems for penetration testing, such as Black Arch, ArchStrike, Kali Linux, and others. [181—183].

The total number of characters that can be used when creating a key is 95 **(Table 7.5)**.

Based on the analysed popular dictionaries, the author formulates the most appropriate combinations of sets based on **Table 7.6**.

Table 7.5 Characters available for key specification in WPA/WPA2 wireless security protocols

No	Character set	Number of characters in the set	Can be used in a dictionary as an atomic unit
1	[0-9]	1	+
2	[a-z]	26	+
3	[A-Z]	26	+
4	Special characters (`~!@#$ %^&*()+-=/|<>[ ]'".,?:;{})	32	+
5	Space	1	-

Table 7.6 Combination of symbols based on Table 3.5

No	Combination of symbols	Number of symbols in the set
1	[0-9] + [a-z]	3
2	[0-9] + [A-Z]	36
3	[a-Z] + [A-Z]	52
4	[0-9] + [a-Z] + [A-Z]	62
5	[0-9] + [a-Z] + [A-Z] + special characters	94
6	[0-9] + [a-Z] + [A-Z] + special characters + space	95

In order to gain access to the network, the attacker must intercept the handshake packet and start the decryption process. Password decryption is performed using the central or graphics processors.

After the dictionary has been searched and the password has not been found, it is obvious that the next dictionary should be used. If the dictionary remains the same in size, but the number of characters for key generation increases, it is advisable to exclude combinations that are already contained in previous dictionaries (7.9):

$$D = \sum_{i=1}^{n} d_i = I_i - I_{i-1}. \tag{7.9}$$

Where D is the general dictionary, d is the current dictionary, and I is the set of keys in the dictionary (7.10):

$$I_i = \{k_1, k_2 \ldots k_n\},$$

where k is the key.

If the base dictionary does not contain the required key, then a new dictionary must be generated. After the base dictionary, it is obvious that the next dictionary should be the one that provides the minimum key length and the smallest set of characters from Table 3.5. This could be a dictionary consisting only of numbers. As can be seen from formula 3.9, keys that have already been used in the base dictionary should not be reused.

If a dictionary with 108 key variants generated from the minimum set of characters has been processed, the key will not be found, then the next dictionary will be generated with 109 keys and the same minimum set of characters (i.e., a combination of characters from 000000000 to 999999999).

This requires 10 times more computing resources to search for keys in the same amount of time. Alternatively, instead of increasing the key size, you can increase the number of characters in the set, i.e. change the options in **Table 7.6**.

The time spent on selecting a key from the dictionary can be determined by the formula (7.11):

$$t_{BF} = \frac{C_d}{S},$$

where C_d is the number of keys in the dictionary, and S is the search speed obtained using the aircracking tool with the S flag.

In order to bypass Wi-Fi protection with the WPA2 wireless security protocol, in addition to the necessary knowledge, an attacker must have the necessary hardware and software for computing. A brute force attack on a password is carried out using a central processing unit (CPU) or graphics processing unit (GPU). The speed of the GPU in performing a brute force attack is significantly higher, but not every computer is equipped with a discrete graphics card. To speed up the brute force attack, this task can be performed in a distributed manner in a computer cluster.

The level of security of the SP, which simulates a specific system, should not be significantly higher than the level and capabilities of the attacker expected to interact with it. In other words, the more complex the key, the more powerful the computing technology required.

Before selecting the next dictionary, the evaluation system must choose the criterion that should be changed in the next dictionary — the length of the keys or the number of characters from which the keys can be generated. We suggest prioritising the dictionary in which the average value of the sum of the percentage ratios of the number of characters and the key length in the following dictionaries is less than the two options (7.12):

$$P_k = \begin{cases} \frac{X_{k+1} + W_i}{2}, X_{k+1} + W_i < X_k + W_{i+1} \\ \frac{X_k + W_{i+1}}{2}, X_{k+1} + W_i > X_k + W_{i+1} \end{cases}. \quad (7.12)$$

The dictionary on which the attack ends will be considered the point for assessing the level of skill and technical equipment of the attacker.

This approach will help to quickly assess the level of security of wireless networks that use WPA/WPA2 wireless security protocols.

Let us introduce the concept of a computing unit, which we will consider to be a specific resource that must perform the operation of brute-forcing one dictionary d_i.

As already mentioned, distributed computing can be used for a frontal attack. Today, various types of computer virtualisation are used to optimise the use of computing resources. These include full, partial, para-virtualisation, and OS-level virtualisation. Let us consider two types of virtualisation: full virtualisation and operating system-level virtualisation, also known as containerisation.

According to research by IBM [184], KVM virtualisation technology on the SUSE Linux Enterprise 11 operating system increases overall resource consumption by 15%. Also, when using virtualisation, the additional costs in processor time consumption are 3-4% higher than without virtualisation.

Containerisation technology does not use a hypervisor, which reduces the load on the hardware. All containers run on the server kernel alone. A separate, isolated environment is created for each container.

As mentioned above, any isolation technology incurs additional costs. In the case of containerisation, these costs range from 0.1% to 1%, due to the use of simple transformations. For example, PID process isolation is performed by adding a 4-byte identifier that indicates which container the process belongs to [185–187].

Fig. 7.7 shows the fundamental difference between containerisation and virtualisation.

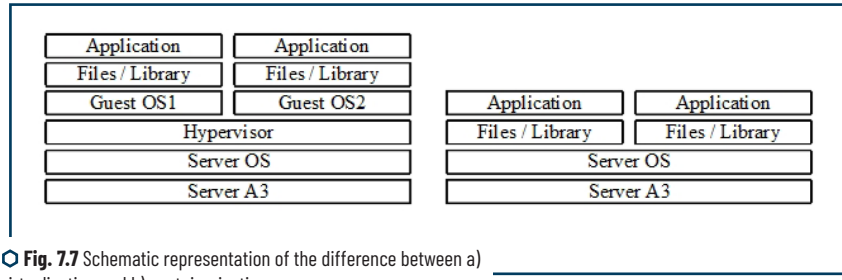

Fig. 7.7 Schematic representation of the difference between a) virtualisation and b) containerisation

To verify the conditional security of SP, data processing speed is of great importance, and therefore preference will be given to technology that can handle dictionary search tasks faster.

To assess the complexity of overcoming WPA/WPA2 wireless security protocols, we will evaluate it using the coefficients of the hierarchy analysis method (HAM) based on criteria such as key length and the number of possible characters in the dictionary. This method allows us to obtain a ratio on a comparison scale with a small deviation, as described in source [178]. The coefficients are used based on actual measurements or subjective opinion. The output is a weight ratio and a consistency index (7.4).

The sums of the coefficients for each column (7.5) are found from the matrix N for further normalisation of the matrix N (7.6).

By summing the coefficients in each row of the normalised matrix and dividing the resulting sum by the number of coefficients in the row, as shown in formula (7.7), the weight of each evaluated element can be determined.

The sum of all weights must be equal to 100%, and therefore, to obtain the percentage value of the complexity of a key of a certain length, its weight must be added to the sum of the previous ones.

7.4 APPLICATION OF CLOUD COMPUTING TO DETERMINE THE SECURITY LEVEL OF DECOY SYSTEMS AND IEEE 802.11 WIRELESS NETWORKS

There are two main approaches to penetration testing: using White-Box and Black-Box methods. In the case of White-Box testing, the auditor has full or partial access to the infrastructure and other components that need to be interacted with during testing [188]. When using the Black-Box method, the security

auditor evaluates the organisation's network infrastructure from a distance and does not have a complete understanding of the internal technologies used.

When modelling the behaviour of an attacker, we assume that all tests are performed in Black-Box mode. Since the Pre-Shared key is encrypted, the attacker has only one option — a frontal attack on the captured associative packets. This may seem ineffective, but it is worth remembering that for this type of attack, it is not necessary to be close to the TD. The attacker may only need significant computing resources for a brute force attack or dictionary attack. Thus, our goal is to demonstrate the real scale of the threat to any organisation.

It is also important to note that to intercept a handshake packet, an attacker does not necessarily have to wait for a new device to connect to the network. Some wireless adapters, when using non-standard drivers, can send reassociation packets to the network, which will interrupt network connections and initiate a new key exchange between clients and the TD. In this case, to intercept the necessary packets, the attacker needs at least one client to be connected to the network. For a successful attack, the attacker must be close to the AP so that their adapter and antenna have enough power to send disassociation packets and intercept handshake packets.

Within the scope of this work, a software and hardware complex was developed that simulates the actions of a potential attacker [189]. **Fig. 7.8** shows a schematic diagram of the components and interactions between the modules of the complex that intercept and decrypt keys for networks operating with the WPA2-PSK wireless security protocol of the IEEE 802.11 standard.

Before starting operations with the complex, it is necessary to install a device for monitoring radio frequency airwaves and intercepting packets in a location where the signal level from the network and its clients is sufficient for interception [190]. The minimum signal level is usually around -90dBm. After installing and activating the interceptor, commands can be sent from the control centre, for example, via SMS, which activates the handshake packet interception procedures (step 1 in **Fig. 7.8**).

Such a message may include data about the TD from which the handshake packets need to be intercepted, the type of interception (active using deauthentication packets or passive), etc. After that, the interceptor switches to scanning mode. The information from such a message is transmitted to the GSM/GPRS module and immediately sent to the computer for further processing (step 2 in **Fig. 7.8**). The Wi-Fi adapter is switched to monitoring mode with settings that allow filtering all other data from the radio frequency airwaves (step 3 in **Fig. 7.8**).

After detecting the required handshake packet, the computer receives a file with the .cap extension (step 4 in **Fig. 7.8**). The data from the .cap file is prepared by the computer for further transmission via the GSM channel (step 5 in **Fig. 7.8**), after which it is transmitted to the command centre (step 6 in **Fig. 7.8**). The command centre then sends the processed .cap file to the decision-making centre (step 7 in **Fig. 7.8**), which in turn starts searching the database for known passwords (step 8.1 in Fig. **Fig. 7.8**). If the key is not found at the previous stage, the decision-making centre creates several virtual computers in the cloud environment and transfers the .cap file for a further brute force attack (step 8.2 in **Fig. 7.8**). After the key is successfully detected, it is transferred to the decision-making centre (steps 9.1, 9.2 in **Fig. 7.8**), from where it is returned to the command centre (step 10 in **Fig. 7.8**).

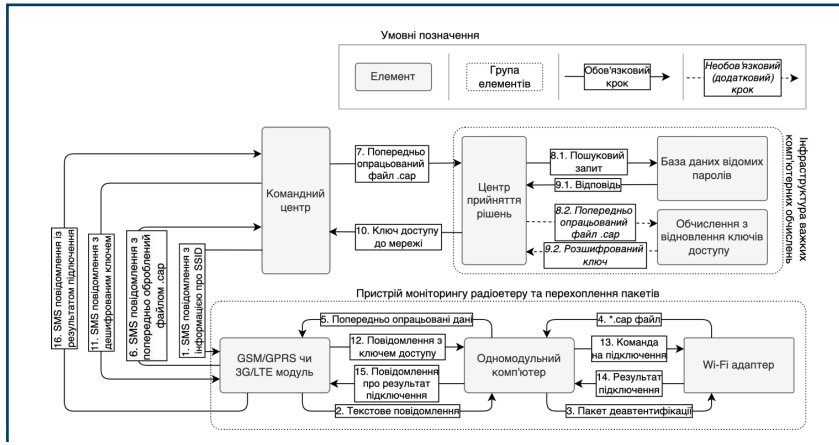

Fig. 7.8 Schematic diagram and communication sequence between the interception and key decryption complex modules

The command centre generates an SMS message with the access password to the specified TD and sends it to the GSM/GPRS interception module (step 11 in **Fig. 7.8**). Data from the GSM/GPRS module is transmitted to a single-module computer (step 12 in F**Fig. 7.8**). The computer generates a command to connect to the Wi-Fi network (step 13 in **Fig. 7.8**). Regardless of whether the connection is successful, the computer receives the result from the network card (step 14 in **Fig. 7.8**). This result is processed by the computer, a message is generated, which is then transmitted to the GSM/GPRS module (step 15 in **Fig. 7.8**). The result data is sent to the command centre (step 16 in **Fig. 7.8**). In the event of a positive result from the interception (step 14 in **Fig. 7.8**), the command centre switches off the computing power and goes into standby mode or restarts the entire process in the event of a negative response. The next attempt to intercept a handshake packet between the same client and the TD will not work, as it is likely that the previously intercepted packet was invalid, i.e. it did not contain the correct access key to the TD.

Environments that store data about the vulnerabilities of certain systems often become the target of attacks by malicious actors, as a successful attack on one link can provide access to information about the vulnerabilities of other systems or even to computing resources as needed [191, 192]. **Fig. 7.9** and **7.10** show the architecture of a heavy computing link based on the Amazon Web Services (AWS) cloud provider. As shown earlier in **Fig. 7.8**, the last link in the complex includes elements responsible for key computation. It is logical to place such resources in private network segments to prevent access from the Internet. However, if these elements need Internet access, they can use Network Address Translation (NAT) gateways located in public segments of the virtual private network. Thus, requests will be routed through the Internet access gateway (**Fig. 7.9**).

The most secure access to elements from the Internet can be provided using Virtual Private Network technology (**Fig. 7.10**).

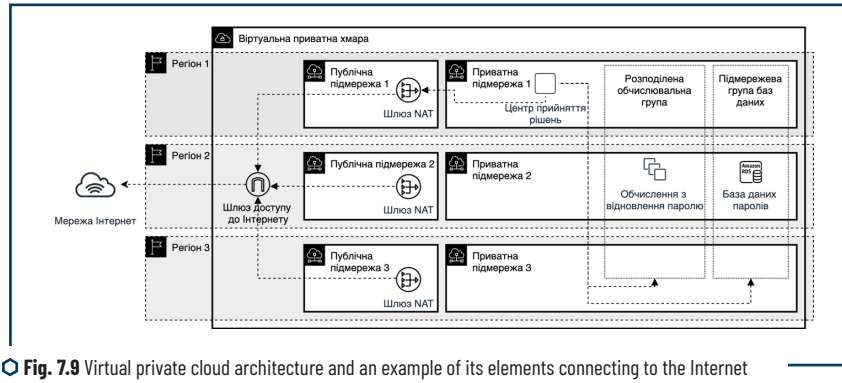

Fig. 7.9 Virtual private cloud architecture and an example of its elements connecting to the Internet

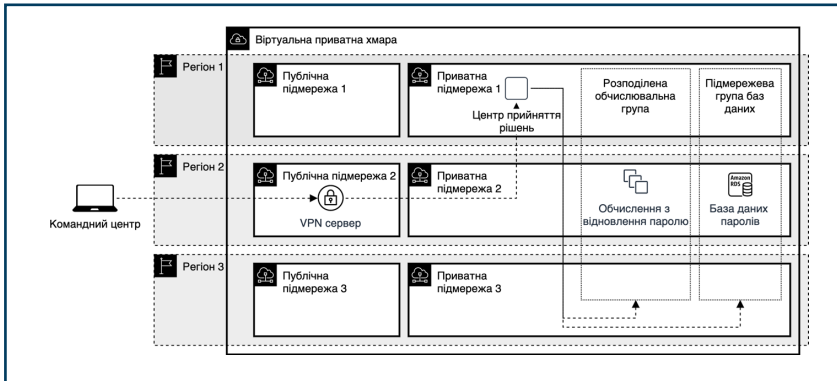

Fig. 7.10 Architecture of a virtual private cloud and an example of communication between the command centre and its elements

The VPN server, located in the public segment, waits for a connection, and when a client connects to it, a network configuration is provided that allows the client to communicate with the same network nodes as the VPN server. After that, in this case, the command centre can interact with the resources of the virtual private cloud without hindrance.

7.5 IMPROVING METHODS FOR DETECTING INTRUSIONS IN IEEE 802.11 NETWORKS USING ARTIFICIAL INTELLIGENCE SYSTEMS

The first version of the IEEE 802.11 standard did not declare a single wireless security protocol. One of the mechanisms that provided protection from unauthorised users was client filtering by MAC address.

Although network equipment manufacturers are constantly adding new security mechanisms, there is still no single mechanism that can fully guarantee the security of the network and its clients.

Each device belonging to the IEEE 802.1x family of standards has unique identifiers assigned by manufacturers. These identifiers are very convenient to use to filter access to client devices. When MAC address filtering is enabled, clients whose MAC addresses are not included in the list of allowed addresses will not be able to connect to the network. However, attackers have tools that allow them to bypass MAC address filtering, and MAC address spoofing can only be detected during investigations.

Specifics of MAC addresses in Wi-Fi networks . As mentioned above, by default, each Wi-Fi client device stores data about the networks to which it has been connected. This is because client devices reconnect immediately after entering the coverage area of access points. This feature makes life easier for users, as they do not need to enter authentication data each time. However, the main problem is that a large amount of metadata about the networks to which the client has been connected can be intercepted and processed by attackers. Such metadata is available in plain text at the channel level of the OSI model.

If a client device is connected to any network, it will actively transmit data to and receive data from the AP. Of course, most of this data is encrypted, but administrative data such as the MAC address of the AP and client, the AP channel, the AP wireless security protocol, and the AP name (SSID) are available for viewing if the network card is set to monitoring mode. If the client is not connected to any network, the SSID data to which the client was connected may be available, as the client device is constantly searching for them.

It is natural for an attacker to use administrative data and other metadata to carry out attacks on the client and the infrastructure as a whole.

If an attacker knows the client's MAC address and the SSID of the AP to which the client is connected, they can spoof an AP with the same MAC address and SSID, but without any wireless security protocol, even if it was used on the legitimate access point. Increasing the signal strength of the fake AP and a DoS attack can help the attacker disconnect the client from the legitimate AP and connect it to the cloned AP. This type of attack is called "Evil Twin".

In another modification of the above attack, the attacker creates a clone of the AP and redirects all client requests to a phishing page where the client is asked to enter their password to supposedly confirm its authenticity.

Once the client has entered their password, the attacker immediately receives it, and the server providing the "evil twin" service is shut down.

If the client device is not connected to any TD, it will actively search for networks to which it was previously connected. At the channel level of the OSI model, this process can be identified by the attacker. Using data from client devices, the attacker can create fake TDs.

As long as the legitimate client and the attacker are connected to the same network, the latter can intercept and/or modify the former's traffic.

The rapid development of any technology often brings security issues that are not so easy to spot in the early stages. Once the underlying technology has gained a large number of functional features that have already become widespread among users, fixing security issues can affect overall performance. This approach can cause inconvenience for both manufacturers and end users.

IEEE 802.11 networks allow their users to connect to networks automatically after they have been authorised. Let's imagine that this functionality will be disabled for security reasons. First of all, each client will be forced to authorise themselves every time they want to use any of the access points. The WDS mechanism or other network mechanism for distributed network access will become useless, since when moving between APs, the user will be forced to authorise each time.

7.5.1 DEVELOPMENT OF A METHOD FOR IDENTIFYING MAC ADDRESS SPOOFING AND "EVIL TWIN" ATTACKS

Before sensors can identify the positioning and behaviour of customers, they must identify each other's positioning. The main problem is that in monitoring mode, sensors cannot identify each other because they cannot transmit data in this mode. Therefore, before starting monitoring mode, each sensor must generate a special frame that will be sent to neighbouring sensors. Using the WHaaS model, once the sensors are in place, a conditional button must be pressed, after which the process of exchanging service data will begin. In addition, such frames cannot be created according to a specific template in order to avoid creating signatures for attackers who already know about the existence of such SP and SBB, and the principle by which synchronisation between sensors occurs [193].

Considering the above, the following synchronisation method is proposed, based on the creation of synchronisation packets in the WHaaS model command centre. When the command centre receives a request from a user, it begins preparing data for the sensors. For example, these can be beacons used by TDs to notify clients of their presence, or search packets used by client devices to search for the networks to which they are connected. In the next step, the command centre sends out data that is unique to each sensor. External elements receive a list of commands and, according to the queue, send the corresponding data to the air at a specific, randomly determined interval. In parallel with the process of sending their data, the sensors listen to the air and identify their neighbours using a filter from the previously received data list. If an external element receives data from its neighbour, it must immediately transmit the identifier of the element whose signal it received and the power of its signal. At the end of the synchronisation process, the end user receives a map with the location of the complex's elements and must confirm or deny the correctness of its construction.

If the method of identifying external elements is based on the transmission of beacons, as any TD does, additional masking should be applied, since the authenticity of the TD can be determined using its MAC address. The database of manufacturers' MAC addresses can be used to avoid suspicion on the part of an attacker regarding the identity of the system he plans to attack. The first 24 bits of the MAC address can tell the attacker about the manufacturer of the device he has identified on the air. In addition to the MAC address, a standard SSID can also be used.

To identify neighbours, it is enough to intercept only one frame from the air, and this can also be a marker for an attacker. Therefore, another parameter that must be generated randomly is the number of frames that will be transmitted to the air.

An example of infrastructure with reference coverage of a system for detecting attacks related to MAC address spoofing for IEEE 802.11 networks . Let us assume that the object to be monitored is located in a room that is a perfect square. In order to achieve better coverage, the sensors must be placed equidistant from each other. Let us illustrate this scheme under ideal conditions (**Fig. 7.11**). The schematic representation of the coverage area shown in **Fig. 7.11** is a simple example of how the angulation method can help in detecting intrusions. If it is known that the user should be within the range of three specific sensors and should not be within the range of others, then this process can help detect an intruder, for example, if the appearance of a client device on the air is detected by sensors within whose range it should not appear. However, a challenge arises in a situation where an intruder uses, for example, a narrow-beam antenna and the power of their signal does not exceed the sensors within which it should operate, then such an attack will not be detected.

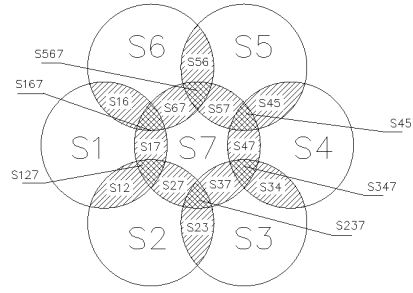

Fig. 7.11 Example of the placement of intrusion detection system sensors for an IEEE 802.11 network

For the diagram shown above, the truth table presented in **Table 7.9** will be valid.

Table 7.9 Truth table for the diagram shown in Fig. 7.10

Sensor identifier	S1	S2	S3	S4	S5	S6	S7
S1		+	–	–	–	+	+
S2	+		+	–	–	–	+
S3	–	+		+	–	–	+
S4	–	–	+		+	–	+
S5	–	–	–	+		+	+
S6	+	–	–	–	+		+
S7	+	+	+	+	+	+	

7.5.2 APPLICATION OF MACHINE LEARNING IN STUDYING THE BEHAVIOUR OF IEEE 802.11 NETWORK USERS AND SUBSEQUENT INTRUSION DETECTION

Currently, there is no method that can help identify MAC spoofing attacks in IEEE 802.11 networks. However, this problem can be partially solved with the help of machine learning.

Each client Wi-Fi device has its own unique daily path, which depends on its owner. Otherwise, the device may be statically located, for example, it may be a router. Therefore, different approaches must be applied to access points and client devices.

Signal quality and stability depend on a large number of external factors. The signal can be negatively affected, for example, by neighbouring access points operating on the same frequency channel, household electrical appliances, interior walls and furniture. Fig. 7.12 shows a simulation of the signal strength with the circuit shown in **Fig. 7.12**.

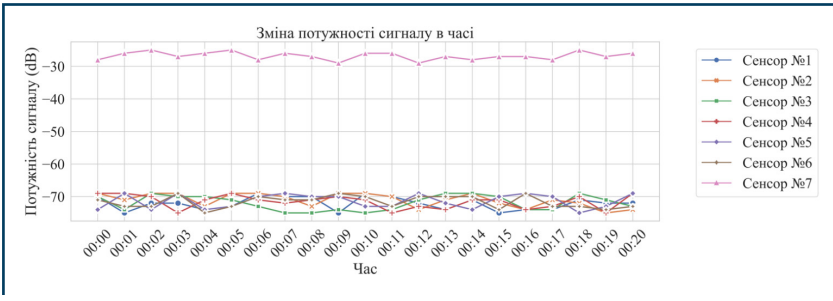

Fig. 7.12 Variation in the signal strength of a legitimate Wi-Fi device over time

Fig. 7.12 shows that the signal strength from sensor No. 7 significantly exceeds the others, which may indicate that the client device is in close proximity to the above-mentioned sensor. Accordingly, the signal strength of the device being monitored is stable on all other sensors and equidistant from the sensor near which this device is located.

In a "malicious twin" attack, the attacker creates a TD that duplicates the legitimate one in terms of basic registration data. For an ordinary user, the TD created by the attacker will not arouse any suspicion. However, there is a high probability that even if the TD is perfectly cloned, it will physically be located in a different place. This means that the signal strength of such a TD will be fundamentally different. **Fig. 7.13** shows the signature of the appearance of an "evil twin".

As can be seen in **Fig. 7.13**, although the signal strength from the legitimate device is preserved on sensor No. 7, the signal strength from the observed device increases between the tenth and eleventh time points on sensor No. 4.

The location of access points is mostly stationary. This indicates that the implementation of artificial intelligence systems based on the signal strength of Wi-Fi access points is an achievable goal, since the neural network or classifier does not need to be retrained frequently.

Unlike access points, studying client behaviour is a much more difficult task, as client devices appear and disappear along with their owners. Here is a list of metadata that can be used to study client behaviour in order to avoid MAC spoofing attacks and others:

Determining the time when a client device appears and disappears in the coverage area. It is very important to know the usual behaviour of a client device, for example, the time when a client device appears and disappears from the coverage area. For example, let's describe the typical behaviour of an employee of a hypothetical company. From Monday to Friday, the employee arrives at the office at 9 a.m., leaves at 1 p.m. and returns at 2 p.m., stays at work until 6 p.m. and leaves until the next morning. Of course, a person cannot arrive at exactly the same time every day and follow the same pattern of behaviour due to the influence of other people and factors, but with such data, it is very easy to identify abnormal behaviour. An example of abnormal behaviour would be a customer appearing in the coverage area on a Saturday night, for example.

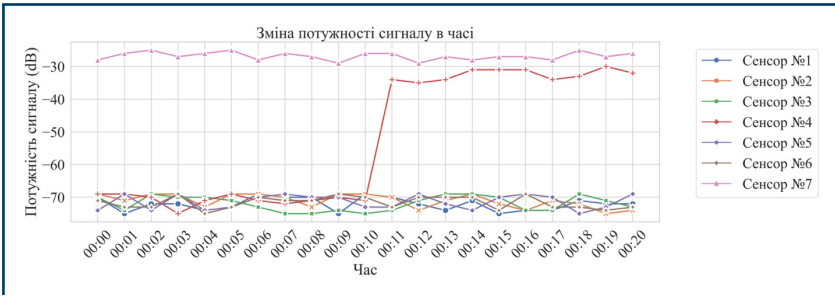

Fig. 7.13 Variation in signal strength of legitimate and illegitimate Wi-Fi devices

Detection of a narrow-beam or other powerful antenna. If all or almost all sensors identify a signal from a client device of a certain client, the highest signal strength is detected by sensors located at a certain outer edge of the infrastructure, and the strength gradually decreases in the opposite direction, this may indicate a possible attack in which the attacker is using a narrow-beam antenna (**Fig. 7.14**).

Based on the scenario described and the sensor placement diagram in **Fig. 7.11**, **Fig. 7.14** shows the variation in signal strength of a potential attacker with a directional antenna. **Fig. 7.14** shows that the highest power is recorded by sensor No. 3, followed by sensors No. 7, No. 2, No. 4 and No. 6, which corresponds to the through-the-wall propagation of signal power.

Identification of a duplicate MAC address in the coverage area. If another device with a MAC address that is already registered in the network at a certain point in time appears in the coverage area, this may indicate an attack. Based on the scenario described above and the infrastructure shown in **Fig. 7.11**, **Fig. 7.13** shows the signal strength variation of a potential attacker who has launched a MAC spoofing attack alongside a client already registered on the network.

Fig. 7.13 can be interpreted as follows: the first sensor to identify the client was sensor No. 3, as evidenced by the change in signal strength (time interval between 0:00 and 0:03). Subsequently, the movement vector changes towards sensor No. 7, which is adjacent to sensor No. 3, indicating natural movement within

the coverage area (time interval between 0:03 and 0:06). Then, in addition to sensors No. 3 and No. 7, the client device is identified by sensor No. 2. The device changes its location from sensor to sensor, and at 0:17, sensors No. 1 and No. 5, which are not adjacent, detect signals from devices with the same MAC address.

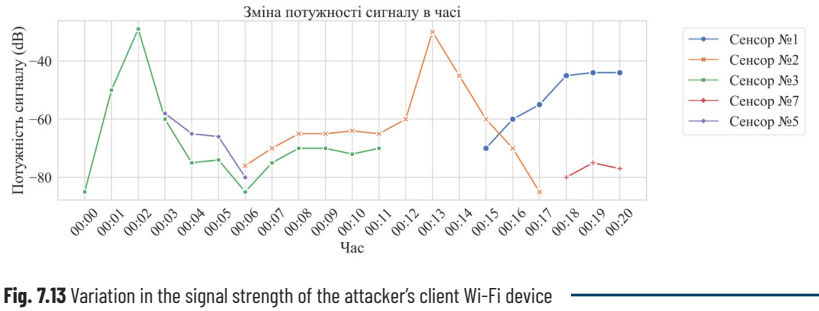

Fig. 7.13 Variation in the signal strength of the attacker's client Wi-Fi device (with a directional antenna) over time

Detection of a client device in a place where it definitely should not be. If the building or network to be protected is large, it is likely that there should be restrictions on access to certain rooms. Detection of client devices in unusual locations (locations covered by certain sensors) may indicate the presence of an intruder who has physically occupied a convenient location or malicious actions on the part of personnel.

Detection of MAC Spoofing attacks using search packets from client devices. Every Wi-Fi client device that has already been connected to a number of wireless networks will search for them using special search packets. This means that each client device will have its own list of access points, which it will actively search for. In any case, even if an attacker changes the MAC address of their device to a legitimate one, they will send search packets with values that are not characteristic of a legitimate device, or not send them at all. Such activities can serve as a marker of an attack.

7.5.3 USING MACHINE LEARNING TO DETECT INTRUSIONS IN IEEE 802.11 NETWORKS

This section has already provided an example of signatures that can indicate a range of attacks involving the substitution of Wi-Fi devices in the air using signal strength analysis. However, building a system to protect against attacks on Wi-Fi networks based solely on signature rules is a major challenge for network administrators, as any changes in the room will affect the signal strength from the access point to the user device or sensor performing the monitoring.

There are quite a few different approaches to implementing machine learning models, which can be supervised or unsupervised. Supervised learning usually requires less computing power to train the model.

Since the data collected from the air may contain certain noise, i.e. the signal level from the TD may sometimes be unstable, it was decided to use the k-nearest neighbours (KNN) classification algorithm for

data analysis in this work. KNN classifies the received data point based on its proximity to other data points in the training set. In this case, model training can be applied to the maximum, minimum, and average signal strengths, as well as the signal strength that appeared most often.

To calculate the distance between two data points, a distance metric is used. Commonly used distance metrics include Euclidean distance, Manhattan distance, and Minkowski distance [194].

In training our model, the Minkowski method was taken as the distance metric. The Minkowski metric is a generalised form of other distance metrics, such as Euclidean distance and Manhattan distance. The Minkowski distance of order p between two points is determined by the formula (7.13):

$$D(x,y)=[(\sum_{i=1}^{n}|x_i-y_i|^p)^{(1/p)}], \quad (7.13)$$

where x and y are two data points with features, p is a parameter that determines the order of the distance metric, and $D(x,y)$ is the Minkowski distance between x and y.

When $p=1$, Minkowski's metric reduces to Manhattan distance, and when $p = 2$, it reduces to Euclidean distance. In general, a larger value of leads to a stronger emphasis on larger differences between feature values and a weaker emphasis on small differences. For training the model, the distance metric order was set to 2, and therefore the distance metric is equivalent to the Euclidean distance (7.14):

$$d=\sqrt{((x_1-x_2)^2+(y_1-y_2)^2)}. \quad (7.14)$$

Among the nearest neighbours, the number of data points belonging to each class is counted. The class with the largest number of nearest neighbours is assigned to the invisible data point. The value determines the number of nearest neighbours and can be selected based on experimentation. In our , the value 3 was selected experimentally.

Majority voting: when nearest neighbours are found, the data point is assigned to the class with the majority of nearest neighbours. If is an odd number, the class with the largest number of nearest neighbours is the majority.

If is an even number, the majority class is determined by considering the values of the nearest neighbours relative to the data point [195].

In mathematical terms, the majority vote can be expressed using 7.15:

$$y=majority(y_1,y_2,\ldots,y_K), \quad (7.15)$$

where is the class label of the nearest neighbours, and the majority returns the class label that occurs most frequently among the nearest neighbours.

The sequence of the classification process is shown in **Fig. 7.15**.

Let us give a brief description of the classification process shown in **Fig. 7.15**:

— the input block represents the input data for the KNN algorithm, which consists of a data set consisting of data points and a query point.

— Step 1 involves setting the value , which determines how many nearest neighbours to consider when making predictions.

— Step 2 involves calculating the distance between the query point and each data point in the dataset .

— Step 3 involves selecting data points from the dataset that have the shortest distance to the query point .

— Step 4 involves counting the number of data points in the selected set that belong to each class and selecting the class with the largest number as the predicted class for the query point .

— The output block represents the output of the KNN algorithm, which is the predicted class for the query point .

The integrated system consists of three layers. The first is the information collection layer. This layer consists of three components: a data collection device, a data aggregator, and a time series database. The task of this layer is exclusively data collection. The second is the data analysis layer. This layer consists of a repository of data prepared for training, a computing environment for training, and a component for continuous integration and delivery of the machine learning model. The third is the application level. This layer consists of a web traffic load balancer, an application that operates on a trained model, and a database that stores data about detected attacks.

The complex works as follows: in the first stage, sensors collect data. After collection and aggregation, the data is recorded in a database based on time series. Once a sufficient amount of data has been collected, it is sent to to the data warehouse for training the machine learning model, after which the model is trained. Next, the trained model is delivered to the server application.

After the process of collecting, training, and delivering artefacts, the components from the information collection level are set to standby mode, the purpose of which is to monitor the signal level and verify the data with the model. If necessary, you can call up the mode of re-collection and re-training or retraining of the model.

8 IMPROVING MECHANISMS FOR IDENTIFYING INTRUSIONS, THE MALICIOUS PERSON, AND THE EFFECTIVENESS OF DECOY SYSTEMS IN IEEE 802.11 WIRELESS NETWORKS

8.1 RESEARCH WITHIN THE FRAMEWORK OF DEVELOPING A METHODOLOGY FOR TRACKING ATTACKERS

The object of the research was a device that had previously been connected to more than a dozen public and private Wi-Fi networks. The main goal was to intercept search packets from this device and find possible locations where the owner of the client device had previously been. The research will allow us to analyse how accurate the information from public databases on the geolocation of Wi-Fi access points is.

Identification of shortcomings in the WiGLE public database in the context of the study . At the same time, both an advantage and a disadvantage of public platforms for searching for the geolocation of Wi-Fi access points is that anyone can upload data to them. On the one hand, this allows more information to be collected, but on the other hand, it raises questions about the quality of this data [89]. Also, the data cannot be checked or verified by the owners of such platforms. Users who upload data collect and process it using non-standardised methods and various software and hardware complexes, which leads to errors in database queries. This study used the public WiGLE geolocation database, as it is the database with the largest amount of data, since the service has been storing historical data since 2001.

The limitation of the WiGLE platform at the moment is that information about each AP can be recorded every millisecond, and each such record will be considered unique, but it is impossible to select aggregated data only for a specific access point regardless of time. This means that there can be so many records for the same AP that even in a city such as Lviv, searching for a single AP can take more than a week. Since the Dot11ProbeReq packet does not transmit MAC address information, but only the SSID identifier, the search has to be based on only one parameter, and the uniqueness of the data in this case is questionable.

One of the records found on the device that simulated the attacker's device was a record about a TD with the identifier Tenda_716370, whose location was known. This network identifier was selected among others as one that has a random set of numbers and is highly likely to be unique in a given territorial unit.

In an area of approximately 2 square kilometres, after the first request, the WiGLE database returns more than 100 unique records after the first request for a TD with the identifier Tenda_716370, which is a rather dubious result.

After more than 20,000 queries to the service, there are already 135 unique access points with the studied identifier. And assuming the data is valid, this suggests that data selection and aggregation fall on the shoulders of the research engineer (**Fig. 8.1**).

Considering that the researcher has no information about the MAC address, such information can be equated to noise, the filtering of which also requires significant computational and human resources.

The visualisation of data in Fig. 8.1 may also indicate a problem with the standardisation of user equipment used to collect information or its invalidity. And considering that the researcher has no information about the MAC address, the probability of determining the geolocation of the attacker's previous locations tends to zero. For the WiGLE platform and similar services, it is important that the user has a GPS module

and a Wi-Fi network card. However, there is no analysis of the type of Wi-Fi adapter antenna used by the user, its power, or the accuracy of the GPS module.

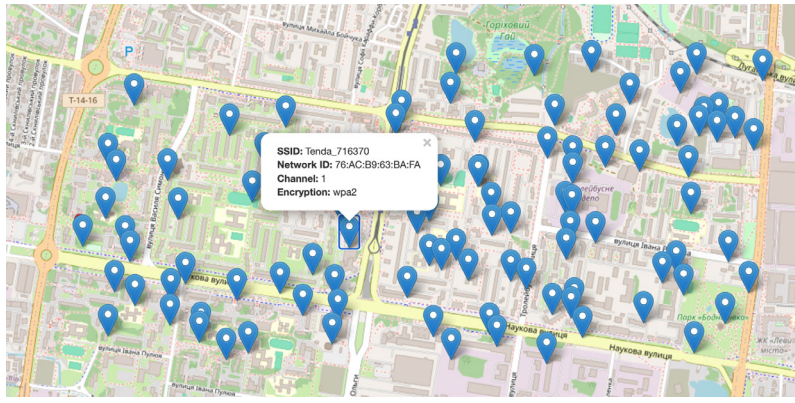

Fig. 8.1 Visualisation of data obtained from the WiGLE database about the access point with the identifier Tenda_716370

Since we can only know one parameter from the Dot11ProbeReq packets intercepted from the attacker's device, namely the SSID network identifier, the information about the location of the TD with a specific name must be as accurate as possible. Since the WiGLE database returned quite a lot of data about the TD named Tenda_716370 in an area of 2 km, the validity of such data should be questioned.

Verification of data obtained from the WiGLE database and application of the power-based approximation method . Since the validity of the data obtained is questionable, it became necessary to conduct a search for access points with the identifier Tenda_716370 within the studied area. In order to improve the accuracy of the geolocation of access points, it became necessary to create our own software.

The Scapy library, which has no alternative in processing network packets in the Python programming language, was used to intercept packets. The logic of the programme consists of continuous monitoring of the ether. If the programme encounters a packet with data containing the Dot11ProbeReq layer, such a packet is sent for further processing. Each unique MAC address found is recorded in a dictionary, where the MAC address is the key and the value is a list of unique names (SSID) of access points that were present in the Dot11ProbeReq packets and are related to the MAC address of the client device.

The data was collected in a specific area of approximately $^{2\ km^2}$ using a computer with an additional GPS module and an Alfa AWUS036NHA network card, which can operate in monitoring mode (**Fig. 8.2**).

The use of a GPS module for this task is mandatory, since when a new AP is detected, its current coordinates must be recorded. The Alfa AWUS036NHA network card was chosen because it is external and its antenna can be replaced as needed. Ultimately, any other network card that can operate in monitoring mode can be used for this task.

Based on the need to improve the accuracy of access point geolocation, a proprietary algorithm was developed to search for Wi-Fi access points, which was then used to collect information (**Fig. 8.3**).

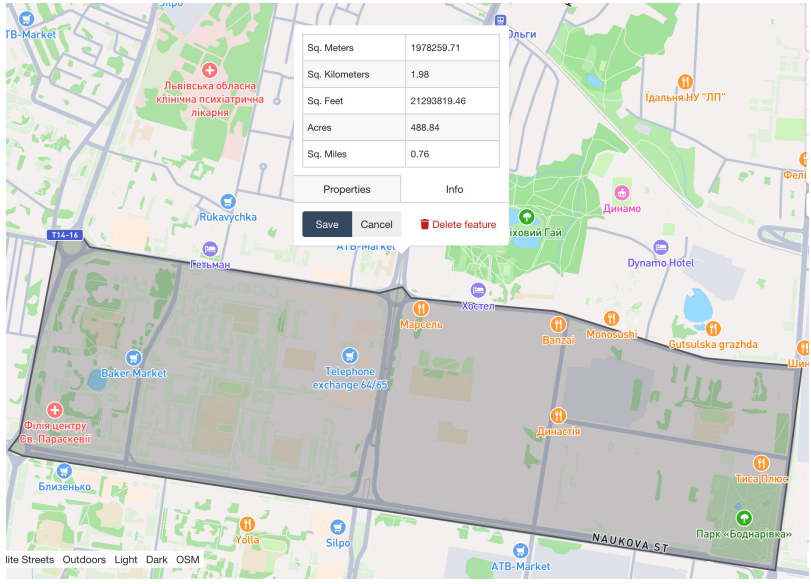

Fig. 8.2 Area where the Tenda_716370 access point was searched for in the WiGLE service

This algorithm allows the collection and aggregation of data related to access points that carry unique information. Each access point is assigned a unique identifier (uuid) based on its MAC address and SSID identifier. If information about an AP appears for the first time, data about its MAC address, SSID, signal strength, channel, and geolocation are recorded in the local database.

If the AP data is already available in the database, the information about the signal strength, channel, and geolocation is rewritten only if the signal strength is higher than the previous one. Thus, by applying the power-based approximation method, it is possible to achieve accuracy in detecting the position of the desired resource.

As a result, the developed approach, using the same perimeter (**Fig. 8.3**), made it possible to collect information about 17,037 access points, of which 13,950 records are intact (those where the GPS module received data from the satellite). Duplicate networks with the same SSID were also found, which may be due to the fact that the TDs belong to the same system and operate in mesh mode, or have the same name and belong to different owners (**Fig. 8.4**).

The OpenStreetMap project, which is free and distributed under the Open Database License, was used to visualise the data on the map. **Fig. 8.5** visualises all detected APs in a given territorial unit.

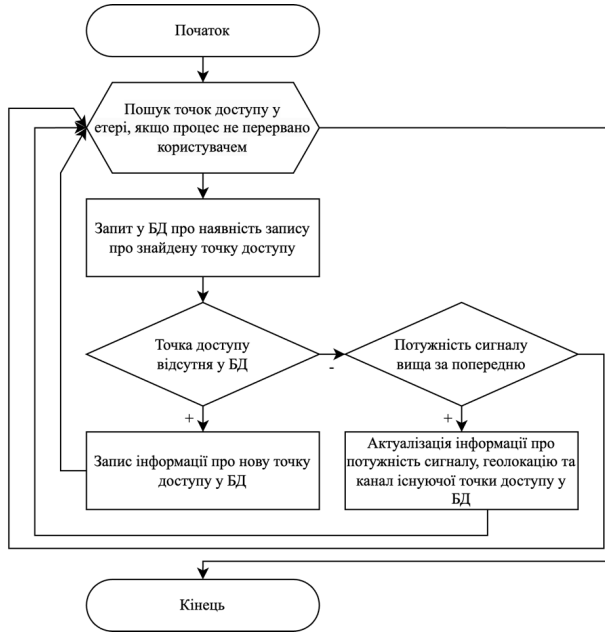

Fig. 8.3 Algorithm for collecting and aggregating geolocation data for Wi-Fi access points

```
sqlite> SELECT * FROM location_data LIMIT 10;
50:d4:f7:ac:f6:4c|Struk|-80|1|49.7855425|24.0251378333333
d4:da:21:5f:de:a6|Peremoha|-63|1|49.7854881666667|24.0250241666667
da:da:21:5f:de:a6|<hidden>|-66|1|49.7855065|24.0244428333333
b0:95:75:26:fb:ee|TP-Link_FBEE|-64|1|49.7854295|24.0245796666667
b0:4e:26:a2:b1:f2|ARKA|-83|1|49.7851281666667|24.0246595
14:eb:b6:ad:54:c7|TWINS|-83|2|49.7854361666667|24.0245905
00:31:92:b9:8f:f6|str115v-97|-75|3|49.7855601666667|24.0245703333333
74:4d:28:4e:c6:44|R I M|-80|3|49.7853183333333|24.0246308333333
06:31:92:b9:8f:f6|<hidden>|-68|3|49.785696|24.0244365
50:ff:20:2f:05:28|Stealth|-33|3|0.0|0.0
sqlite> SELECT COUNT(*) FROM location_data;
17037
sqlite> SELECT COUNT(*) FROM location_data WHERE latitude != 0.0 AND longitude != 0.0 ;
13950
sqlite> SELECT COUNT(DISTINCT ssid) FROM location_data;
12393
sqlite> SELECT COUNT(DISTINCT ssid) FROM location_data WHERE latitude != 0.0 AND longitude != 0.0;
10355
```

Fig. 8.4 Analysis of collected data from the radio ether on the geolocation of Wi-Fi access points

Filtering by SSID parameter and latitude and longitude intervals was added to the visualisation. Thus, we can enter the identifier for the interval we are interested in and compare the accuracy of the public WiGLE database of collected data with the current data collected using the presented approach (**Fig. 8.6**).

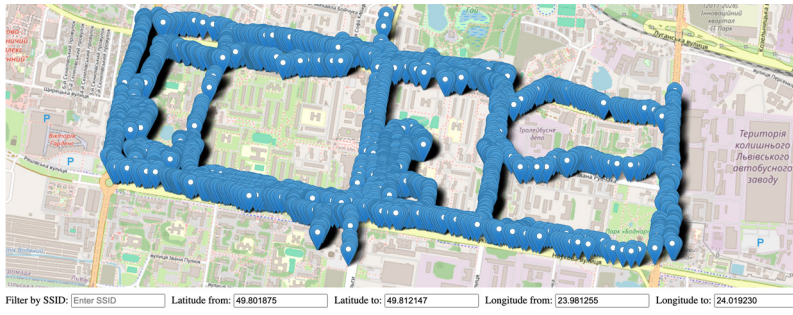

Fig. 8.5 Visualisation of data on Wi-Fi access points on a map

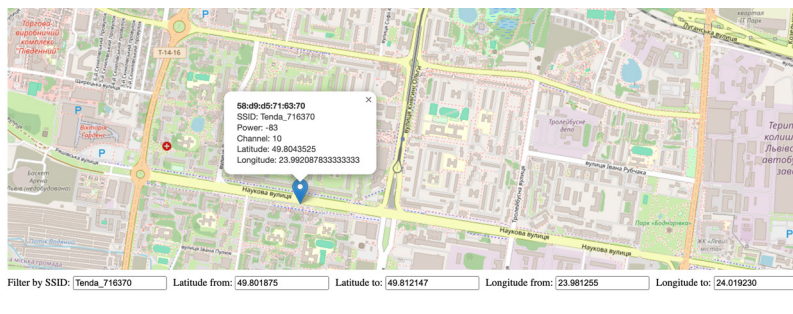

Fig. 8.6 Searching for a specific access point and visualising it on a map

From **Fig. 8.6**, we can see that the wireless network identifier Tenda_716370 is unique for the studied perimeter of 2 km^2. If we compare the data obtained from the public WiGLE database with the available data, we can see that it is quite difficult to obtain the exact coordinates of the location of a specific device, even in a small area. Therefore, we can conclude that data aggregation and the power-based approximation method allow us to obtain a more accurate geolocation of the device we are looking for.

8.2 APPLICATION OF A DIAGNOSTIC MODEL OF A DECOY SYSTEM FOR IEEE 802.11 WIRELESS NETWORKS

Based on formula (7.4) and **Table 7.4**, we will find the weight matrix of each of the elements in relation to each other:

N=‖(1&2&V5&V8&V9&V7@V2&1&V5&V8&V9&V7@5&5&1&V6&V9&V6@8&8&6&1&V4&4@9&9&9&4&1&8@7&7&5&V4&V8&1)‖.

We will find the sums of the coefficients for each of the columns for further normalisation of matrix N (7.5).

S_1=1+1/2+5+8+9+7=30.5;

S_2=2+1+5+8+9+7=32;

S_3=1/5+1/5+1+6+9+5=21.4;

S_4=1/8+1/8+1/6+1+4+1/4=5.7;

S_5=1/9+1/9+1/9+1/4+1+1/8=1.71;

S_6=1/7+1/7+1/6+4+8+1=13.45.

According to formula 3.6, we will find the normalised matrix |N|:

|N|=‖(1/30.5&2/32&(1⁄5)/21.4&(1⁄8)/5.7&(1⁄9)/1.71&(1⁄7)/13.45@(1⁄2)/30.5&1/32&(1⁄5)/21.4&(1⁄8)/5.7&(1⁄9)/1.71&(1⁄7)/13.45@5/30.5&5/32&1/21.4&(1⁄6)/5.7&(1⁄9)/1.71&(1⁄6)/13.45@8/30.5&8/32&6/21.4&1/5.7&(1⁄4)/1.71&4/13.45@9/30.5&9/32&9/21.4&4/5.7&1/1.71&8/13.45@7/30.5&7/32&5/21.4&(1⁄4)/5.7&(1⁄8)/1.71&1/13.45)‖=‖(0.033&0.062&0.009&0.022&0.065&0.011@0.016&0.031&0.009&0.022&0.065&0.011@0.164&0.156&0.047&0.029&0.065&0.012@0.262&0.25&0.28&0.175&0.146&0.297@0.295&0.281&0.421&0.702&0.585&0.595@0.23&0.219&0.234&0.044&0.073&0.074)‖.

By summing the coefficients of each row of the normalised matrix and dividing the sum by the number of coefficients in the row using (7.7), we obtain the weight of each of the protection mechanisms.

x=[((0.033 + 0.062 + 0.009 + 0.022 + 0.065 + 0.011)/6@(0.016 + 0.031 + 0.009 + 0.022 + 0.065 + 0.011)/6@(0.164 + 0.156 + 0.047 + 0.029 + 0.065 + 0.012)/6@(0.262 + 0.25 + 0.28 + 0.175 + 0.146 + 0.297)/6@(0.295 + 0.281 + 0.421 + 0.702 + 0.585 + 0.595)/6@(0.23 + 0.219 + 0.234 + 0.044 + 0.073 + 0.074)/6)]=[(0.034@0.026@0.079@0.235@0.48@0.146)].

According to the matrix x, we obtain the coefficients for each of the protection mechanisms. Filtering by MAC address is equal to 0.034; hiding the SSID by disabling the function of sending beacons into the open air is equal to 0.026; the WEP wireless security protocol is equal to 0.079; the WPA/WPA2 wireless security protocols are equal to 0.235; WPA3 is equal to 0.48; WPS is equal to 0.146.

Based on the coefficients obtained as a result of calculations using the hierarchy assessment method, we can estimate the complexity of circumvention measures for each combination of IEEE 802.11 wireless security mechanisms by adding these coefficients on a scale from 0 to 1.

Since the enabled WPS mechanism only worsens the security of the TD, the combination of this mechanism with WPA2 wireless security protocols will differ in terms of the difference in coefficients, namely:

WPA2-WPS= 0.235-0.146=0.089.

In order to attract the attention of attackers, protection at the access point must definitely be present, but in such a way that it can be overcome with relatively little effort.

8.3 ANALYSIS OF RESULTS ON IMPROVING THE METHODOLOGY FOR DISTRIBUTED SELECTION OF ACCESS KEYS TO THE WPA2 PROTECTION MECHANISM IN IEEE 802.11 NETWORKS

The Kali Linux operating system was used to derive the basic vocabulary for a brute force attack. Kali Linux is a Linux-based operating system based on the Debian distribution, which is free and freely available. The direct purpose of this operating system is penetration testing. However, in addition to testing their own systems, it is possible that such operating systems could be used by attackers for attacks [196, 197].

As part of the experiment, we analysed the dictionaries that come with the Kali Linux operating system and are located in the /usr/share/wordlists directory. These dictionaries are filtered to exclude all word combinations that are less than 8 characters long and those that do not consist entirely of ASCII characters. These dictionaries are integrated into one, from which non-unique word combinations are excluded:

cat dirbuster/.txt fern-wifi/* rockyou.txt fasttrack.txt metasploit/*.lst metasploit/*.txt | grep ..... | grep -P -v «[^[:ascii:]]» | sort —unique*

Of course, each test depends on the type of hardware and software configuration. This study investigated the speed of key calculation for the WPA2 wireless security protocol from a previously intercepted handshake packet.

The research was conducted in a CoreOS virtual machine container, which was allocated 1 GB of RAM and one Intel Core i5-4590 processor core with a clock speed of 3.3 GHz.

Preliminary testing was performed using the aircracking — S command, which allows measuring the speed of a brute force attack in terms of keys per second.

After executing the command and counting the number of keys, we obtain the value *Cd* = 9801317.

The speed of the brute force attack was measured 10 times for two different approaches, virtualisation and containerisation. The results are presented in **Table 8.1** and visualised in **Fig. 8.7**.

Table 8.1 Measurements of the speed of key brute force attacks using a dictionary

Attempt type of virtualisation	1	2	3	4	5	6	7	8	9	10
Full virtualisation (keys/second)	922	956	954	955	927	911	949	947	933	912
Containerisation (keys/second)	1053	1038	1038	1041	1031	1032	1038	1038	1054	1036

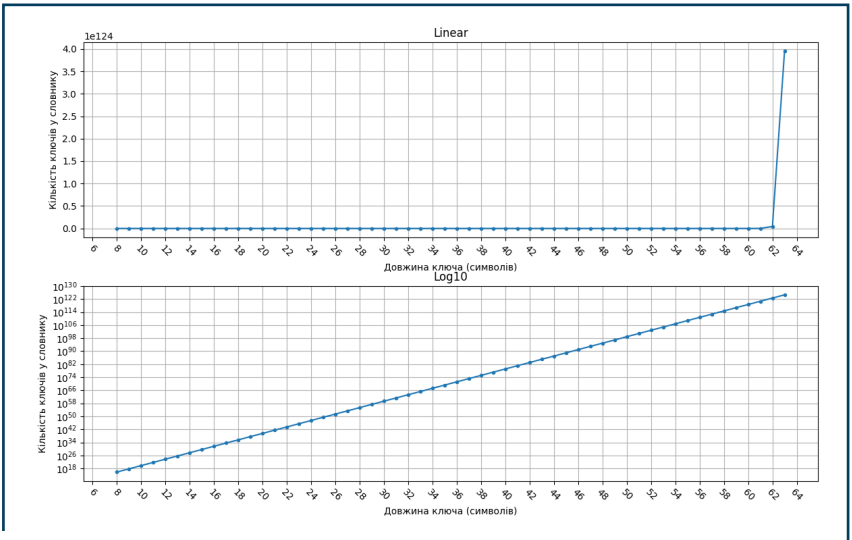

Fig. 8.7 Visualisation of the comparison of key search speeds using virtualisation and containerisation

Let's calculate the average speed for virtualisation:

(S_V) = (922+956+954+955+927+911+949+947+933+912)/10=936,6.

Let's calculate the average speed for containerisation:

(S_V) = (1053+1038+1038+1041+1031+1032+1038+1038+1054+1036)/10=1039,9.

Knowing the number of keys in the base dictionary and the speed at which the keys are processed, we can use formula (7.11) to find the approximate time it will take to brute force the entire base dictionary.

t_BF = 9801317/1039,9 ≈ 9425 s ≈ 2 hours 37 min.

If the key is not found in the base dictionary, the key search will start from the eight-character dictionary. As mentioned above, WPA2 allows you to set a key length from 8 to 63 characters, which is equivalent to 56 key length options. This means that the number of options will be increased tenfold when switching to the next dictionary ( ), and therefore ten times more power will be required, or ten times more time will be spent with the same power.

Searching for an access key from an intercepted handshake packet is a resource-intensive operation. Although the number of all possible keys is finite, it is still quite large and can be calculated using formula (8.1).

$$n=\sum_{i=8}^{63} 95^i. \tag{8.1}$$

As the key length increases, the size of the dictionary increases exponentially (**Fig. 8.8**).

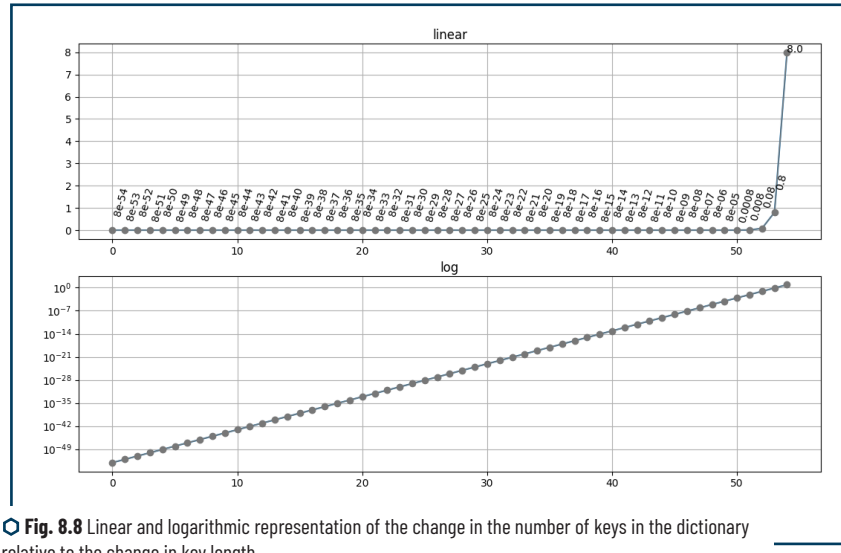

Fig. 8.8 Linear and logarithmic representation of the change in the number of keys in the dictionary relative to the change in key length

The data in **Fig. 8.8** show that with each new iteration, ten times more computing power is required, which in turn is very difficult to implement without cloud computing.

Despite the results obtained, it is important to note that when using different hardware, the results will be similar to others. However, in this study, this will not affect the weight coefficients obtained as a result of the hierarchy analysis method, since it is based on the difference between reference points, which in this study are represented by static dictionaries.

To apply the hierarchical analysis method to this study, it is necessary to evaluate the characteristics in relation to each other on a scale from 1 to 9. Since the complexity of the dictionary increases linearly, i.e., tenfold each time (**Fig. 8.9**). Accordingly, on the segment from 1 to 9, the cost of transition from one dictionary to another will be the addition of a coefficient of 0.145454545 to the previous value.

The object of the study is IEEE 802.11 standard networks, which are inherently vulnerable to ZD attacks. The main hypothesis of this study is the possibility of identifying the appearance of illegitimate Wi-Fi TD alongside legitimate ones. During the development of the SVM model, it was assumed that the attack could be prevented using supervised machine learning algorithms.

The main device used in the study was a Raspberry Pi 4 Model B single-board computer, although any other computer with similar characteristics could be used to reproduce this study. Also, to intercept packets, a network card is required to work in IEEE 802.11 networks with the ability to operate in monitoring.

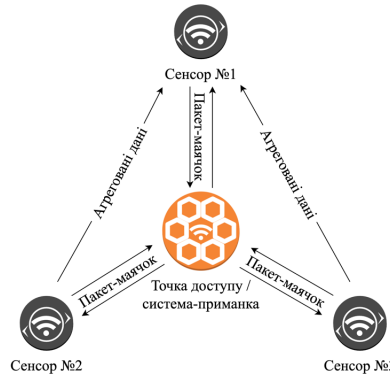

Fig. 8.9 Linear and logarithmic representation of changes in vocabulary complexity in the hierarchy analysis method

That is, the weight of a dictionary with keys 8 characters long is 0.3011%, the weight of a dictionary with keys 9 characters long is 0.3128%, the weight of a dictionary with keys 63 characters long is 5.1451%, and the weight of a dictionary with 64-character keys is 5.3967%.

As already mentioned, key length is not the only criterion for assessing its complexity. The set of characters present in the dictionary is also an important criterion. Based on **Tables 7.5–7.6**, we can draw conclusions about which combinations of characters in dictionaries are logical to use. The maximum number of characters that can be used in a password is 95, i.e. 95 characters constitute 100%. Accordingly, we obtain **Table 8.2**.

Table 8.2 Percentage ratio of the number of characters in dictionaries

Number of characters	**95**	**94**	**62**	**52**	**36**	**32**	**26**	**10**
Percentage representation (w)	100	98.95	65.26	54.17	37.89	33.68	27.36	10.5

8.4 DETECTION OF "EVIL TWIN" ATTACKS ON IEEE 802.11 (WI-FI) NETWORKS USING THE KNN CLASSIFICATION MODEL

An evil twin attack is a type of attack on a wireless network where an attacker sets up a fake Wi-Fi access point with the same name as a legitimate access point to trick users into connecting to it. Once a user connects to the fake AP, the attacker can intercept and manipulate the user's network traffic, steal confidential information, and launch further man-in-the-middle attacks [198].

WPA3 includes a new feature called Simultaneous Authentication of Equals (SAE), also known as Dragonfly, which provides stronger protection against RAT attacks. SAE uses a secure key exchange protocol

to establish a unique encryption key for each session, making it much more difficult for an attacker to intercept and decrypt the user's network traffic.

However, it is important to note that while WPA3 is more resistant to MD attacks, it does not completely protect against them. Since a large number of client devices were manufactured before the invention of the WPA3 protocol, network device manufacturers have built in a transitional mechanism that allows older user devices to work with new network equipment [199]. Subsequently, in 2019, evidence was published that some WPA3 routers were vulnerable to downgrade attacks, which could potentially allow an attacker to bypass SAE protection and perform a "double agent" attack [200].

As of 2021, the global economic value of Wi-Fi was estimated at 3.3 trillion USD, and it is expected to grow to 5 trillion USD by 2025. This technology is undoubtedly a driving force behind modern business. However, Wi-Fi technology, like most modern technologies, also has certain shortcomings in terms of information security. One of the most common attacks on wireless networks is the man-in-the-middle attack. While wired networks require physical access to the premises where the computer network operates to carry out a similar type of attack, wireless networks do not. An attacker can be located at a distance from the access point and easily clone a legitimate network. Using directional antennas, an attacker can make the signal from an illegitimate access point stronger than that from a legitimate one. In this way, attackers can lure clients into a network that they control. User and corporate data can then be stolen or modified.

As we know, every electronic device, even those manufactured at the same factory using identical technology, is unique. Even devices that are identical in terms of hardware and software configuration can still have different digital fingerprints due to variations in electronic components. When it comes to IEEE 802.11 networks, the digital fingerprint of a TD at the OSI model channel level can be the signal strength. In Wi-Fi networks, even the slightest deviation of the router antenna can change the network coverage area. When it comes to a TD attack, it is usually carried out using equipment that is quite different from that used to distribute Wi-Fi traffic. In addition, the location of the equipment plays an important role. If the Wi-Fi TD is located in a certain room and the attack is carried out from outside its boundaries, then at the moment of the attack, the signal level from the TD on a group of client devices will be completely different than in the absence of an attack.

But in today's world, where most customers are mobile and can change their place of work without any restrictions, this cannot be a reliable source for detecting attacks. Such a source could be a device that remains static in one place and periodically checks the signal level from a given TD, then concludes that an attack is present based on the level of the signal received. Usually, one such device may not be enough, since the signal level from a legitimate base station and from a malicious base station may coincide under certain conditions, and then the attack will go unnoticed. Therefore, the more devices that monitor the signal level from the legitimate TD, the higher the probability of detecting an attack, since the attacker has less chance of finding the correct location and signal power configuration on the device acting as the ZD.

Data analysis and intrusion detection are usually quite costly procedures. At the same time, devices that monitor traffic cannot be bulky, at least larger than the network equipment itself, and the price of such a system cannot exceed the price of the information processed in a legitimate network. Therefore, this study presents a method for detecting attacks called ZD, which fully meets all of the above requirements [200].

The object of the study is IEEE 802.11 standard networks, which are inherently vulnerable to ZD attacks. The main hypothesis of this study is the possibility of identifying the appearance of illegitimate Wi-Fi TD alongside legitimate ones. During the development of the SVM model, it was assumed that the attack could be prevented using supervised machine learning algorithms.

The main device used in the study was a Raspberry Pi 4 Model B single-board computer, although any other computer with similar characteristics could be used to reproduce this study. Also, to intercept packets, a network card is required to work in IEEE 802.11 networks with the ability to operate in monitoring mode. The study used an Alfa Network AWUS036NHA network card with an AR9271 chipset, which was selected from the list of network equipment recommended for use by the aircrack-ng software tool, which in turn is based on the Scapy library [200]. The single-board computer was installed with the Linux Raspbian operating system, with a minimal set of additional software to avoid excessive resource consumption. The device being monitored was any Wi-Fi router; in our case, we used a Xiaomi Mi 4A.

The software for intercepting packets from the Wi-Fi air was developed using the Python programming language. The functionality of intercepting packets from the air was implemented using the Scapy library. Hereinafter, such devices will be referred to as sensors.

A database based on InfluxDB time series was installed on the main single-board computer to record data from the sensors.

In a single-sensor setup, software that intercepts packets and the InfluxDB database are also installed on it. In a setup with two or more sensors, InfluxDB is installed on only one of the sensors, and a Wi-Fi access point is deployed on it, which is used to deliver data to the database and for other service communications.

Scapy is a powerful Python library for processing and analysing network packets, namely creating and sending network packets, checking and analysing network traffic, testing and analysing network security. In this study, our programme intercepted packets that contained a beacon (Dot11Beacon layer in Scapy). A beacon in Wi-Fi is a type of control frame that is periodically sent by the AP to announce its presence and provide clients with basic information about the network, and is therefore the best type of packet for monitoring. The purpose of beacons is to allow clients to discover available Wi-Fi networks and provide basic information about the network, such as the network name (SSID), supported data rates, security modes, and the signal strength from the AP in decibels multiplied by milliwatts (dBm).

Beacons are used in the initial stage of Wi-Fi connection and are necessary for the proper functioning of Wi-Fi networks. They provide clients with a way to detect available networks and make informed decisions about which network to connect to. Thus, beacons are an important component of Wi-Fi networks, providing network information and allowing clients to discover and connect to the network.

During the Beacon experiment, packets were collected from the air continuously (up to 10 packets per second). Signal strength is measured in decibels. As mentioned above, the sensors are based on single-board computers, so their computing power is quite limited. Also, communication between sensors may be unstable in some cases for a number of reasons.

The main problem is that every second, a request to write to the database may be sent to the service network. When several small packets are transmitted separately, each packet requires its own header and other control information. These overhead costs can quickly accumulate and take up a significant portion

of the available bandwidth, leaving less room for actual data transmission. Data aggregation can help reduce these overhead costs by combining several packets into one larger packet that requires only a single header and management information. By combining several packets into one larger packet, the time required to transfer data can be reduced, which can help reduce latency and improve the overall performance of the service network. There is also a greater risk of packet loss or corruption due to factors such as noise or interference. By combining several packets into one larger packet, the likelihood of packet loss or corruption is reduced, as the larger packet is more robust and less susceptible to such factors.

In general, data aggregation can help optimise network performance in low-speed networks by reducing overhead, latency, and the risk of packet loss or corruption. This typically improves the experience of network clients and ensures more efficient use of available network resources.

Therefore, in order to avoid overloading the InfluxDB server running on a single-board computer and to avoid loading the service network, it was decided to send the aggregated data once per minute. The general scheme of data collection from the air and communication between sensors is shown in **Fig. 8.10**.

A set of metrics in a one-minute interval — the number of intercepted beacons, maximum, minimum and average signal strength, the signal strength that appeared most often in the air. The average power is calculated as the sum of all powers divided by their number (8.2):

$$pwr\_avg=(sum(pwr))/(number\ of\ packets). \quad (8.2)$$

The signal power that appeared most often is found using the statistical function mode (8.3).

$$pwr\_(most\ frequent)=mode(pwr). \quad (8.3)$$

During the measurements, the TD and sensor(s) were located in a dedicated laboratory with no other electronic devices that could affect the signal level, although the influence of the environment cannot be ruled out, as Wi-Fi devices are very common today. Data was collected only in the 2.4 GHz Wi-Fi radio frequency range (2412 MHz — 2472 MHz).

To reproduce the attack in conditions close to real life, a TD with the same MAC address and network identifier (SSID) was created and placed in locations outside the laboratory where the experiment with the legitimate TD was conducted.

The simulation of the legitimate TD was moved 15 times in neighbouring rooms to the one where the sensors and the legitimate TD were located.

Two groups of experiments were conducted. In the first group, a single sensor was used to collect information about the signal strength from the TD (**Fig. 8.10 a**). In the second group, there were three such sensors. The sensors were placed in a conditional triangle around the legitimate TD, i.e., the so-called triangulation approach was used (**Fig. 8.10 b**).

After collecting the data, we can proceed to training the machine learning model. To do this, we need to perform a number of actions, such as:

— selecting a machine learning algorithm based on the specific task;

— preparing the data, which involves collecting, cleaning, and pre-processing the data. The data must be organised in a format that is suitable for the selected machine learning algorithm;

— dividing the data into training and test data. The training data is used to train the model, and the test data is used to evaluate the model's performance;

— the model is trained based on the training data;

— test data is used to evaluate the machine learning model and its performance. This will help determine how well the model is able to generalise new data;

— if the model performs poorly, you can try to improve it by adjusting the algorithm, changing the model parameters, or changing the data.

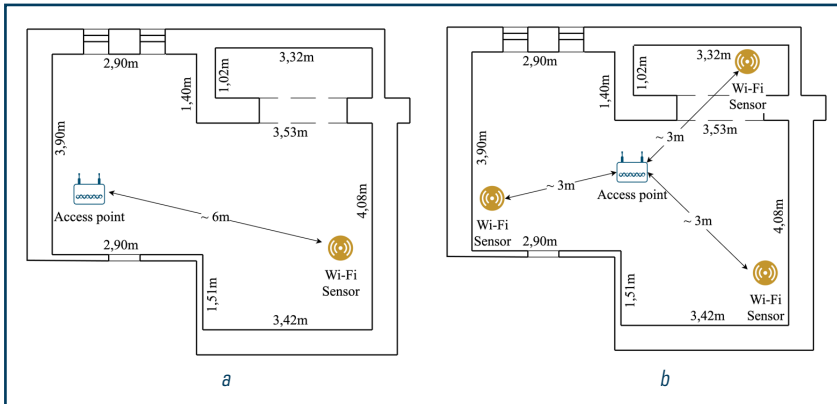

Fig. 8.10. Equipment placement diagram with: *a* — one; *b* — three sensors

If the model meets the requirements set for it, it can be deployed in a working environment to make predictions based on new data.

Research results. A large amount of data is required to train a machine learning model. To understand how ready the collected data is for processing, we visualise small time intervals and compare metrics from legitimate and illegitimate APs (**Fig. 8.11**).

Fig. 8.11 shows the data collected by the same sensor, which reflects the signal strength from legitimate and illegitimate access points over two hours. The figure shows that there is a fairly large gap between the signal strength of legitimate and illegitimate access points. This allows us to conclude that this data can be used for statistical analysis and training a machine learning model.

The Python3 programming language was used for data analysis, with libraries such as numpy for working with multidimensional arrays and matrices, pandas for data manipulation and further analysis, matplotlib for visualising two-dimensional graphs, and seaborn as an extension to matplotlib.

The total number of rows in the datasets was 14,579 for the experiment with one sensor and 20,610 for the experiment with three sensors. To understand how ready the collected data is for model training, we can

call the function to visualise the correlation on a heat map using the seaborn library, which will show how much the data from the dataset correlates with each other (**Fig. 8.12, 8.13**).

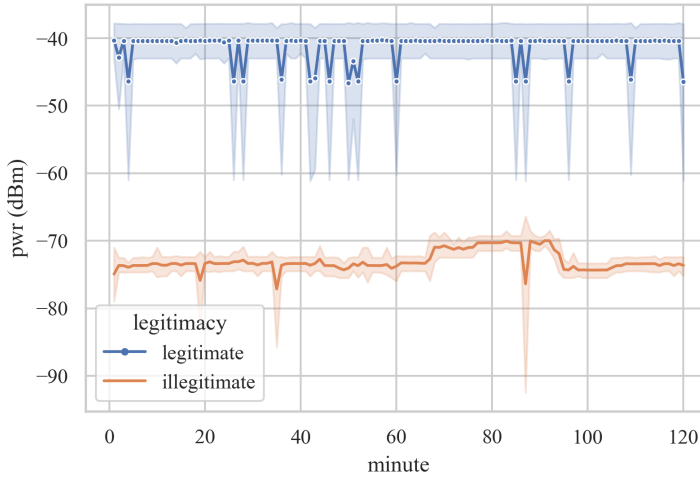

Fig. 8.11 Comparison of signal strengths from legitimate and illegitimate access points

The total number of rows in the datasets was 14,579 for the experiment with one sensor and 20,610 for the experiment with three sensors. To understand how ready the collected data is for model training, we can call the function to visualise the correlation on a heat map using the seaborn library, which will show how much the data from the dataset correlates with each other (**Fig. 8.12, 8.13**).

In **Fig. 8.12**, the metrics *min* — minimum signal level; *max* — maximum signal level; *avg* — average signal level; m_f — mode, or the one that occurs most often; n_p — total number of network packets.

Accordingly, in **Fig. 8.13**, the first part of the metric name s1 — sn corresponds to the sensor number, and the second part, the metric name after the underscore, corresponds to the metrics mentioned in the explanation to **Fig. 8.12**.

As is known, the correlation coefficient measures the strength and direction of the linear relationship between two variables and ranges from −1 to +1, where a coefficient of +1 indicates a perfect positive linear relationship and a coefficient of −1 indicates a perfect negative linear relationship. A coefficient of 0 indicates no linear relationship.

The threshold for "low" correlation may depend on the context and specific application. However, in general, a correlation coefficient of less than 0.3 or greater than -0.3 is often considered a low correlation.

But if there are many variables in the dataset and the correlation between a particular variable and other variables is consistently weak, it may be appropriate to exclude that variable from the analysis or modelling process.

Fig. 8.12 Heat map of metric correlation for a data set with one sensor

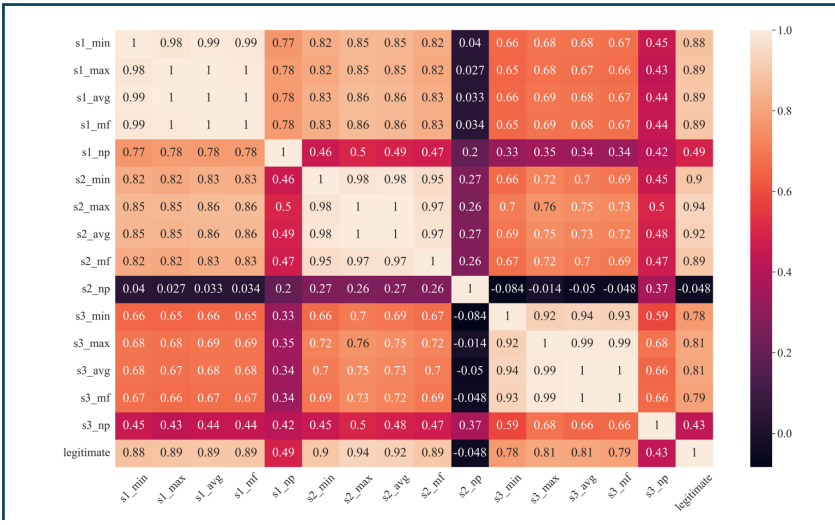

Fig. 8.13 Heat map of metric correlation for a dataset with three sensors

We can immediately see that the columns with the suffix np, which is an abbreviation for number of packets, correlate very poorly with other parameters, and in some cases the values are negative. This means that such parameters can be detrimental to the prediction model. Therefore, they must be disregard-

ed to ensure the accuracy of the model. In **Fig. 8.14, 8.15**, we can see a heat map of metric correlations, but without the metric responsible for the number of intercepted beacon packets.

As can be seen in **Fig. 8.14**, the minimum correlation between metrics is at least 0.45 for the dataset, which is quite good. This may mean that such a dataset can be used to train a machine learning model.

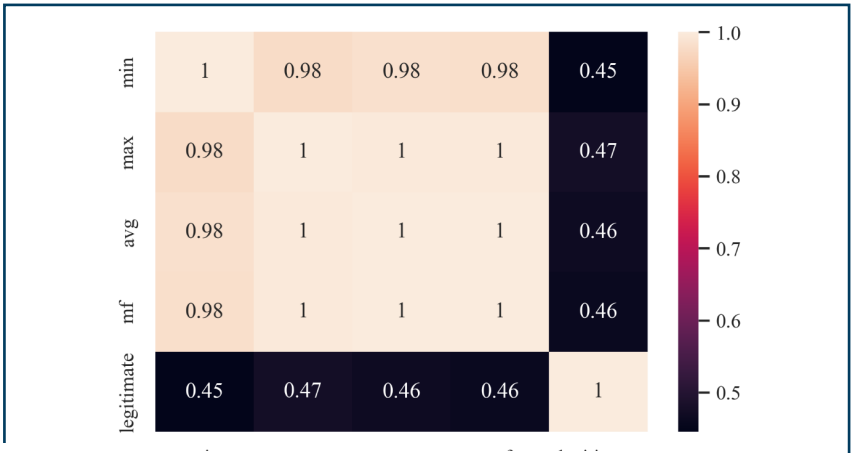

Fig. 8.14 Heat map of metric correlation for a dataset with one sensor without the metric of the number of intercepted beacon packets

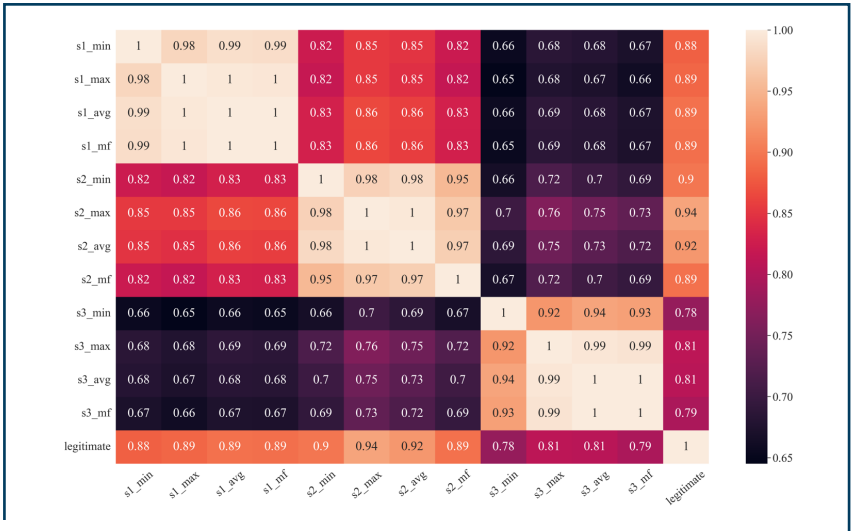

Fig. 8.15 Heat map of data correlation without the metric of the number of intercepted beacon packets

In **Fig. 8.15**, the minimum correlation between metrics is at least 0.65. The lowest correlation value is at the intersection of sensor No. 1 and its most frequent signal strength metric and the minimum signal strength value from sensor No. 3.

As in the case of one sensor, so in the case of three, the smallest correlation value does not fall within the range [– 0.3÷ 0.3], so the data sets can be used to train the classifier.

Now, using the plotpair function of the seaborn library, we visualise the correlation of metrics for a group of experiments with one sensor (**Fig. 8.16**) and for each of the sensors for a group of experiments with three sensors (**Fig. 8.17**, **8.18**). This comparison allows us to clearly see whether there is an intersection of signal strength data from legitimate and illegitimate access points.

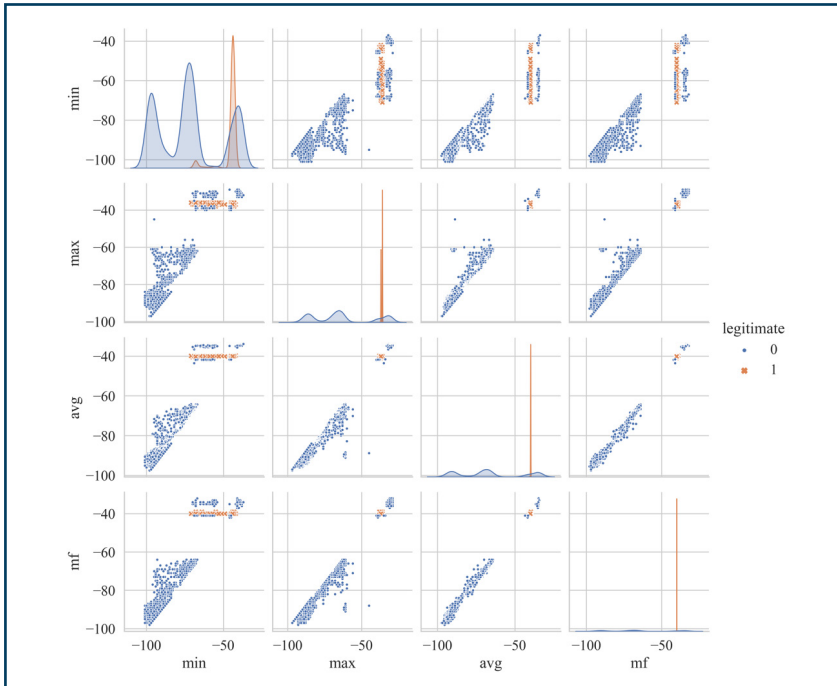

Fig. 8.16. Comparison of metrics for legitimate and illegitimate access points for a group of experiments with one sensor

Fig. 8.16 shows a fairly significant overlap in the minimum signal strength from legitimate and illegitimate access points, which can obviously become a problem when training a machine learning model. However, such a case is quite likely in real conditions, when an attacker places equipment close to a legitimate AP or increases the power of their own equipment in a certain direction, in this case in the direction of the

sensor. This indicates that the use of only one sensor in the context of monitoring a Wi-Fi wireless network may be insufficient.

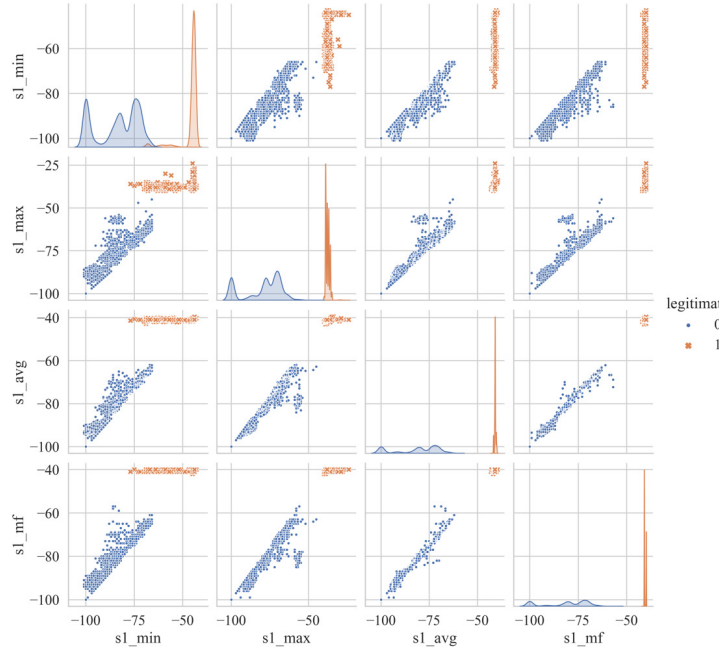

Fig. 8.17 Comparison of metrics for legitimate and illegitimate access points recorded by sensor No. 1

In **Fig. 8.17, 8.18**, for sensors No. 1 and No. 2, respectively, it can be seen that the metrics of legitimate and illegitimate APs do not intersect, which is a good sign for building a machine learning model. In the case of sensor No. 3 (**Fig. 8.18**), as in the case with the single-sensor experiment, there are slight intersections when comparing the metrics of values from legitimate and illegitimate access points.

Despite this distribution, this is quite acceptable, since in real conditions, an attacker can place a "malicious twin" in a location where the sensor will receive packets with a signal level similar to that of the legitimate AP.

To train the model, the following metrics were submitted for the input (X_train): min, max, avg, mf for the experiment with one sensor, and s1_min, s1_max, s1_avg, s1_mf, s2_min, s2_max, s2_avg, s2_mf, s3_min, s3_max, s3_avg, s3_mf for a group of experiments with three sensors. The output metric (y_train) is the legitimate metric, which indicates whether the set of metrics is legitimate. Also, 20% of the entire dataset was used in testing (X_test, y_test). The Scikit-learn library was used to create the machine learning

model. Scikit-learn offers a complete set of machine learning algorithms , including both supervised and unsupervised learning methods.

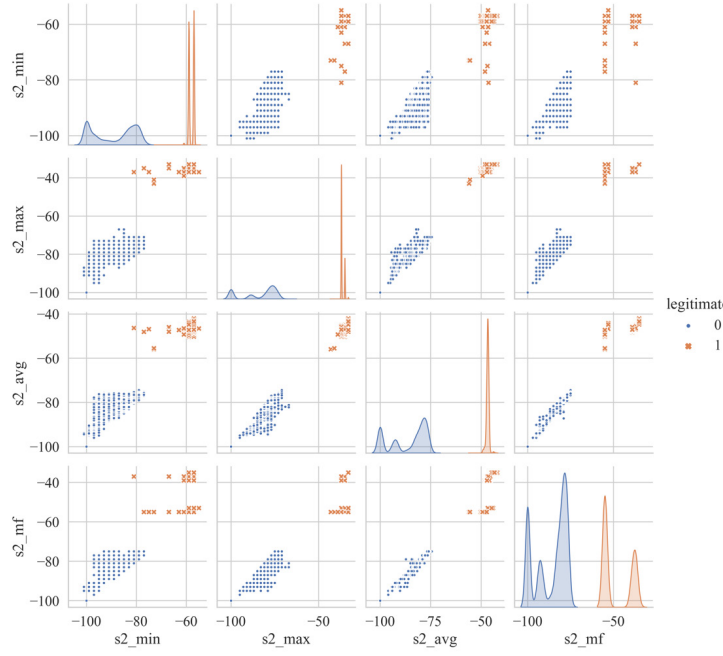

Fig. 8.18. Comparison of metrics for legitimate and illegitimate access points recorded by sensor No. 2

Some of the most commonly used algorithms in scikit-learn are linear and logistic regression, decision trees, random forests, k-nearest neighbours, support vector machines, and neural networks.

After training the model, we conducted tests on twenty percent of the total data set and obtained results for a scheme with one sensor (**Table 8.1**) and three sensors (**Table 8.2**). To do this, we will use the classification_report function from the sklearn library of the metrics class.

Precision is the number of true positive results divided by the total number of positive predictions; *recall* is the number of truly positive samples divided by the total number of actually positive samples; *f1-score* is a harmonic mean of precision and recall, which provides a balance between precision and recall; *support* is the number of actual occurrences of a class in a given dataset. In other words, it is the number of samples in each class.

In addition to precision, recall, f1-score, and support, the classification_report function of the sklearn library also reports three other important metrics: accuracy, macro avg, and weighted avg. Accuracy is the

proportion of correctly predicted labels among all samples in the dataset. This metric measures the overall performance of the model; macro avg is the average value of the metrics (precision, recall, and f1-score) for all classes. This metric is useful when you want to evaluate the overall performance of the model in a multi-class classification task; weighted avg is the value of the metrics (precision, recall, and f1-score) for all classes, weighted by the number of samples in each class. This metric is useful when you have an unbalanced dataset and want to evaluate the overall performance of the model, taking into account the class imbalance.

Table 8.1 Classification report for a single-sensor scheme

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2470
1	1.00	1.00	1.00	446
accuracy			1.00	2916
macro avg	1.00	1.00	1.00	2916
weighted avg	1.00	1.00	1.00	2916

Table 8.2 Classification report for a scheme with three sensors

	precision	recall	f1-score	support
0	1.00	1.00	1.00	2792
1	1.00	1.00	1.00	1330
accuracy			1.00	412
macro avg	1.00	1.00	1.00	4122
weighted avg	1.00	1.00	1.00	4122

As we can see from **Table 8.1**, with 20% of the dataset, namely 2916 cases, the KNN model correctly classified all test cases. We can see the same result in the case of three sensors, but in this case, as you can see, the number of test cases is already 4,122.

Fig. 8.19 a and **b** show the misclassification matrix of the KNN machine learning model for the case with one sensor and three sensors, respectively.

The upper left corner (**Fig. 8.19 a, b**) shows the number of correctly classified illegitimate access points; the upper right corner shows incorrectly classified illegitimate access points; the lower right corner shows the number of incorrectly classified legitimate access points; the lower left corner shows the number of correctly classified legitimate access points.

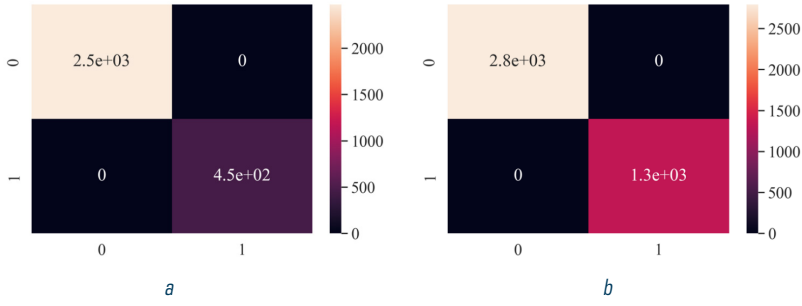

Fig. 8.20. Visualisation of the mismatch matrix for the KNN machine learning model for a scheme with: *a* — one sensor; *b* — three sensors

CONCLUSIONS

The monograph presents a comprehensive study of cybersecurity issues in modern information systems, in particular the detection and counteraction of ransomware, as well as the protection of IEEE 802.11 wireless networks.

Based on the results of the research, the following main conclusions were formulated.

A comprehensive analysis of modern cyber threats, including ransomware viruses and attacks on wireless networks, has been conducted. Models of attackers, their classification and methods of distribution were developed and analysed. Key attributes and characteristics for identifying malicious software were identified, confirming the need to develop new protection methods for both commercial and government information systems.

The eBPF technology for monitoring system calls, file and network activity has been developed and implemented. The created system allows for the effective detection and neutralisation of threats in real time, which has been confirmed by experimental research.

An integrated method for analysing ransomware based on machine learning has been created, combining various approaches (decision trees, support vector machines, deep neural networks) with eBPF technology. The high efficiency of the proposed models has been experimentally proven: SVM methods achieved an accuracy of 94.2% and an F1 score of 92.2%, while decision trees and random forests showed an accuracy of up to 96.5%.

A conceptually new model of information protection system using decoy systems has been developed, which includes a minimum set of necessary elements and rules for their interaction. The use of cloud computing to ensure system scalability has been proposed.

A diagnostic model has been created and implemented to evaluate the effectiveness of IEEE 802.11 wireless network protection mechanisms. A detailed model has been developed for the WPA2 protocol, which allows the decoy system settings to be optimised according to the profile of a potential attacker.

A machine learning model based on the KNN algorithm has been developed to detect "evil twin" attacks in Wi-Fi networks. The model's 100% effectiveness in distinguishing between legitimate and fake access points has been experimentally confirmed, which is critical for protecting against attacks on WPA3.

A methodology for improving the geolocation accuracy of Wi-Fi access points has been proposed, allowing 90-100% accuracy in detecting unique instances, as opposed to 0.5—1% in public databases. The methodology is based on signal strength metrics and standardisation of contributor equipment.

A comparative analysis of virtualisation technologies for calculating the complexity of WPA2 key brute force attacks was conducted. It was found that containerisation is 11% more efficient than full virtualisation while maintaining the ability to quickly scale resources.

Practical recommendations for implementing the proposed solutions have been developed, including methods for optimising machine learning models to reduce system resource consumption and approaches for integrating eBPF modules with existing security systems (SIEM, EDR).

Promising areas for further research have been identified, including the development of adaptive model training methods to counter new types of cyber threats, the improvement of decoy systems, and the creation of integrated solutions for comprehensive information system protection.

The results obtained are of significant practical importance for improving the cybersecurity of modern information systems and can be implemented in both the commercial sector and government institutions.

REFERENCES

1. Kamalrul Bin Mohamed Yunus, Y., Ngah, S. B. (2023). Ransomware: Stages, detection and evasion. 2023 International Workshop on Engineering Technologies and Computer Science (ENT). https://doi.org/10.1109/ICSECS52883.2021.00048
2. Wang, S.-Y., Chang, J.-C. (2022). Design and implementation of an intrusion detection system by using Extended BPF in the Linux kernel. Journal of Network and Computer Applications, 198, 103283. https://doi.org/10.1016/j.jnca.2021.103283
3. Miano, S. (2019). Creating complex network services with eBPF: Experience and lessons learned. 2019 IEEE International Conference on Software Analysis, Testing and Evolution (SATE). https://doi.org/10.1109/HPSR.2018.8850758
4. Hohlfeld, O. (2019). Demystifying the performance of XDP BPF. 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS). https://doi.org/10.1109/NETSOFT.2019.8806651
5. Liu, C. (2020). A protocol-independent container network observability analysis system based on eBPF. 2020 IEEE 26th International Conference on Parallel and Distributed Systems (ICPADS). https://doi.org/10.1109/ICPADS51040.2020.00099
6. Sadiq, A., Syed, H. (2023). Detection of denial-of-service attack in cloud-based Kubernetes using eBPF. Applied Sciences, 13 (8), 4700. https://doi.org/10.3390/app13084700
7. Miano, S., Risso, F., Vaquero, L. M., Sanvito, D., Bianco, A. (2019). Introducing SmartNICs in server-based data plane processing: The DDoS mitigation use case. IEEE Access, 7, 107161–107170. https://doi.org/10.1109/access.2019.2933491
8. Miano, S., Bertrone, M., Risso, F., Vásquez Bernal, M., Lucrezia, M., Piñero, D. R. L. (2021). A framework for eBPF-based network functions in an era of microservices. IEEE Transactions on Network and Service Management, 18 (1), 133—151. https://doi.org/10.1109/tnsm.2021.3055676
9. Chandrakala, D. (2023). Detection and classification of malware. IEEE Transactions on Industrial Electronics. https://doi.org/10.1109/ICAECA52838.2021.9675792
10. Sobesto, B., Cukier, M., Berthier, R., Hiltunen, M. (2011). DarkNOC: Dashboard for Honeypot Management. Proceedings of the 25th Large Installation System Administration Conference (LISA). Available at: https://www.usenix.org/legacy/event/lisa11/tech/full_papers/Sobesto.pdf
11. Smith, J. (2020). Analyzing Honeypot Data Using Kibana and Elasticsearch. Towards Data Science. Available at: https://medium.com/towards-data-science/analysing-honeypot-data-using-kibana-and-elasticsearch-5e3d61eb2098
12. Fan, W., Fernández, D., Du, Z. (2017). Versatile Virtual Honeynet Management Framework. IET Information Security, 11 (1), 38–45. https://doi.org/10.1049/iet-ifs.2015.0256
13. 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

14. Stockman, M., Rein, A., Heile, R. (2015). An Open-Source Honeynet System to Study System Banner Message Effects on Hackers. Journal of Computing Sciences in Colleges, 31 (1), 282–293. Available at: https://www.academia.edu/79459134/An_Open_Source_Honeynet_System_to_Study_System_Banner_Message_Effects_on_Hackers
15. Kumar, A., Kumar, R. (2023). A Highly Interactive Honeypot-Based Approach to Network Threat Analysis. Future Internet, 15 (4), 127. Available at: https://www.mdpi.com/1999-5903/15/4/127
16. Hoque, M. S., Mukit, M. A., Bikas, M. A. N. (2012). An Implementation of Intrusion Detection System Using Genetic Algorithm. International Journal of Network Security & Its Applications, 4 (2), 109–120. Available at: https://arxiv.org/pdf/1204.1336
17. Saeedi, H., Khotanlou, H., Nassiri, M. (2012). A dynamic approach for honeypot management. International Journal of Information Security and Systems Management, 1 (2), 104–109. Available at: https://journals.iau.ir/article_548869_3972fd2299180120e6ed2e4763473ec6.pdf
18. Fraunholz, D., Zimmermann, M., Schotten, H. D. (2017). An Adaptive Honeypot Configuration, Deployment and Maintenance Strategy. 2017 19th International Conference on Advanced Communication Technology (ICACT), 53–57. https://doi.org/10.23919/ICACT.2017.7890056
19. Whyte, C., Mazanec, B. (2023). Understanding Cyber Warfare: Politics, Policy, and Strategy. Routledge. Available at: https://www.routledge.com/Understanding-Cyber-Warfare-Politics-Policy-and-Strategy/Whyte-Mazanec/p/book/9781032159317
20. Humayun, M., Niazi, M., Alshayeb, M. (2019). Cyber Security Threats and Vulnerabilities: A Systematic Mapping Study. Arabian Journal for Science and Engineering, 45, 3171–3189. Available at: https://link.springer.com/article/10.1007/s13369-019-04319-2
21. Kettani, H., Wainwright, P. (2019). On the Top Threats to Cyber Systems. 2019 IEEE International Conference on Information and Computer Technologies (ICICT), 175–179. Available at: https://ieeexplore.ieee.org/document/8711324/
22. Koskinen, A. (2019). DevSecOps: Building Security into the Core of DevOps. University of Jyväskylä. Available at: https://jyx.jyu.fi/handle/123456789/67345
23. Kuvaja, P., Porres, I. (2018). Self-Service Cybersecurity Monitoring as an Enabler for DevSecOps. IEEE Access, 6, 72092–72104. Available at: https://ieeexplore.ieee.org/document/8766805/
24. Jung, S., Won, Y. (2018). Ransomware detection method based on context-aware entropy analysis. Soft Computing, 22 (20), 6731–6740. https://doi.org/10.1007/s00500-018-3257-z
25. Opirskyy, I., Vasylyshyn, S., Piskozub, A. (2020). Analysis of the use of software baits (honeypots) as a means of ensuring information security. Cybersecurity, 2 (10), 88–97. https://doi.org/10.28925/2663-4023.2020.10.8897
26. Zhuravchak, D., Opirskyy, I., Piskozub, A., Dudykevych, V., Tolkachova, A. (2024). Monitoring ransomware with Berkeley Packet Filter. Cybersecurity Providing in Information and Telecommunication Systems. Available at: https://ceur-ws.org/Vol-3550/
27. Zhuravchak, D., Opirskyy, I., Piskozub, A., Dudykevych, V. (2021). Ransomware prevention system design based on file symbolic linking honeypots. 2021 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Cracow, Poland, 22–25 September 2021. https://doi.org/10.1109/IDAACS53288.2021.9660913

28. Bensaoud, A., Kalita, J., Bensaoud, M. (2023). A survey of malware detection using deep learning. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.4363417
29. Kaur, U. et al. (2022). Malware detection using pseudo semi-supervised learning. SpringerLink. https://doi.org/10.1007/978-3-031-09282-4_31
30. Zhuravchak, D., Opanovych, M., Tolkachova, A., Dudykevych, V., Piskozub, A. (2024). Design of an integrated defense-in-depth system with an artificial intelligence assistant to counter malware. Eastern-European Journal of Enterprise Technologies, 6 (2 (132)), 64—73. https://doi.org/10.15587/1729-4061.2024.318336
31. Zhuravchak, D., Dudykevych, V. (2023). Real-time ransomware detection by using eBPF and natural language processing and machine learning. 2023 IEEE 5th International Conference on Advanced Information and Communication Technologies (AICT), Lviv, Ukraine, 1—4. https://doi.org/10.1109/AICT58444.2023.10362535
32. Shemitha, P., unitha Malar Dhas, J. (2023). Trusted detection of ransomware using machine learning algorithms. International Journal of Innovative Technology and Exploring Engineering (IJITEE). https://doi.org/10.35940/ijitee.I1133.0789S219
33. Korobeinikova, T., Zhuravel, I., Mychuda, L., Sikora, A. (2024). The practice of block symmetric encryption for a secure Internet connection. CEUR Workshop Proceedings, 3861, 114—122. Available at: https://ceur-ws.org/Vol-3800/short5.pdf
34. Korobeinikova, T., Tachenko, I., Romanyuk, O., Romanyuk, S., Stakhov, O., Reyda, O. (2024). Assessing network security risks: A technological chain perspective. International Conference on Advanced Computer Information Technologies (ACIT), 565—570. https://doi.org/10.1109/ACIT62333.2024.10712586
35. Vorobets, P., Vakhula, O., Horpenyuk, A., Korshun, N. (2024). Implementing post-quantum KEMs: Practical challenges and solutions. CEUR Workshop Proceedings, 3826, 212—219. Available at: https://ceur-ws.org/Vol-3826/short9.pdf
36. 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. Available at: https://ceur-ws.org/Vol-3504/paper4.pdf
37. Mykhaylova, O., Korol, M., Kyrychok, R. (2024). Research and analysis of issues and challenges in ensuring cyber security in cloud computing. CEUR Workshop Proceedings, 3826, 30—39. Available at: https://ceur-ws.org/Vol-3826/paper3.pdf
38. Mykhaylova, O., Shtypka, A., Fedynyshyn, T. (2024). An Isolation Forest-based approach for brute force attack detection. CEUR Workshop Proceedings, 3842, 43—54. Available at: https://ceur-ws.org/Vol-3842/paper3.pdf
39. Tyshyk, I., Hulak, H. (2024). Testing an organization's information system for unauthorized access. CEUR Workshop Proceedings, 3826, 17—29. Available at: https://ceur-ws.org/Vol-3826/paper2.pdf
40. Stefinko, Y., Piskozub, A., Obshta, A. (2024). Analysis of Vulnerability Characteristics for Automated Penetration Testing. IEEE International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering, 449—453. https://doi.org/10.1109/TCSET64720.2024.10755620

41. 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), 6-20. https://doi.org/10.15587/1729-4061.2023.273363
42. Chen, Q., Bridges, R. A., Skjellum, A. (2019). Automated ransomware behavior analysis: Pattern extraction and early detection. In Science of Cyber Security, 199–214. https://doi.org/10.1007/978-3-030-34637-9_15
43. Or-Meir, O., Nissim, N., Elovici, Y., Rokach, L. (2019). Dynamic malware analysis in the modern era — A state of the art survey. ACM Computing Surveys, 52 (5), 1–48. https://doi.org/10.1145/3329786
44. Almashhadani, A. O., Kaiiali, M., Sezer, S., O'Kane, P. (2019). A multi-classifier network-based crypto ransomware detection system: A case study of Locky ransomware. IEEE Access, 7, 47053–47067. https://doi.org/10.1109/access.2019.2907485
45. Min, D., Choi, S., Lee, S., Park, Y., Kim, S. (2021). A content-based ransomware detection and backup solid-state drive for ransomware defense. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. https://doi.org/10.1109/tcad.2021.3099084
46. Magnani, S., Risso, F., Siracusa, D. (2022). A control plane enabling automated and fully adaptive network traffic monitoring with eBPF. IEEE Access. https://doi.org/10.1109/access.2022.3202644
47. Kumar, N., et al. (2023). AI in cybersecurity: Threat detection and response with machine learning. Tuijin Jishu/Journal of Propulsion Technology, 44 (3), 38–46. https://doi.org/10.52783/tjjpt.v44.i3.237
48. Sanvito, D. (2022). Learning what to monitor for efficient anomaly detection. Proceedings of the 2nd European Workshop on Machine Learning and Systems. https://doi.org/10.1145/3517207.3526979
49. Syrotynskyi, R., Tyshyk, I., Kochan, O., Sokolov, V., Skladannyi, P. (2024). Methodology of network infrastructure analysis as part of migration to zero-trust architecture. CEUR Workshop Proceedings, 3800, 97–105. Available at: https://ceur-ws.org/Vol-3800/short3.pdf
50. Martseniuk, Y., Partyka, A., Harasymchuk, O., Nyemkova, E., Karpinski, M. (2024). Shadow IT risk analysis in public cloud infrastructure. CEUR Workshop Proceedings, 3800, 22–31. Available at: https://ceur-ws.org/Vol-3800/paper2.pdf
51. Deineka, O., Harasymchuk, O., Partyka, A., Obshta, A., Korshun, N. (2024). Designing data classification and secure store policy according to SOC 2 type II. CEUR Workshop Proceedings, 3654, 398–409. Available at: https://ceur-ws.org/Vol-3654/short7.pdf
52. Mykhaylova, O., Fedynyshyn, T., Platonenko, A. (2024). Hardcoded credentials in Android apps: Service exposure and category-based vulnerability analysis. CEUR Workshop Proceedings, 3826, 206–211. Available at: https://ceur-ws.org/Vol-3826/short8.pdf
53. Miano, S., Risso, F., Vásquez Bernal, M., Sanvito, D., Piñero, D. R. L. (2018). Creating complex network services with eBPF: Experience and lessons learned. 2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR). https://doi.org/10.1109/hpsr.2018.8850758
54. Abranches, M., Rocha, R., Pedrosa, L. (2021). Efficient network monitoring applications in the kernel with eBPF and XDP. 2021 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Heraklion, Greece, 9–11 November 2021. https://doi.org/10.1109/nfv-sdn53031.2021.9665095

55. Sadiq, A., Syed, H., Alazab, M., Venkatraman, S. (2023). Detection of denial of service attack in cloud-based Kubernetes using eBPF. Applied Sciences, 13 (8), Article 4700. https://doi.org/10.3390/app13084700
56. Zhuravchak, D., Opirskyy, I., Piskozub, A., Dudykevych, V., Tolkachova, A. (2024). Monitoring ransomware with Berkeley Packet Filter. Cybersecurity Providing in Information and Telecommunication Systems. Available at: https://ceur-ws.org/Vol-3550/
57. Garfinkel, T. (2004). Ostia: A delegating architecture for secure system call interposition. Network and Distributed System Security Symposium (NDSS). Retrieved May 9, 2024. Available at: https://xenon.stanford.edu/~talg/papers/NDSS04/abstract.html
58. Volckaert, S., Coppens, B., De Sutter, B. (2024). System call interposition without compromise. Proceedings of the 2024 IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), 30–41. https://doi.org/10.1109/DSN58291.2024.00030
59. Zhuravchak, D., Opirskyy, I., Piskozub, A., Dudykevych, V. (2021). Ransomware prevention system design based on file symbolic linking honeypots. 2021 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Cracow, Poland, 22–25 September 2021. https://doi.org/10.1109/idaacs53288.2021.9660913
60. Khan, M., Alqahtani, S. (2023). AI-driven threat detection in cloud computing: A survey on honeypot-based approaches. Future Generation Computer Systems, 137, 189–205. https://doi.org/10.1016/j.future.2023.04.015
61. Sagirlar, G., Carminati, B., Ferrari, E. (2018). AutoBotCatcher: Blockchain-based P2P Botnet Detection for the Internet of Things. arXiv preprint arXiv:1809.10775. Available at: https://arxiv.org/abs/1809.10775
62. Commey, D., Hounsinou, S., Crosby, G. V. (2024). Strategic Deployment of Honeypots in Blockchain-based IoT Systems. arXiv preprint arXiv:2405.12951. Available at: https://arxiv.org/abs/2405.12951
63. Golomb, T., Mirsky, Y., Elovici, Y. (2018). CIoTA: Collaborative IoT Anomaly Detection via Blockchain. arXiv preprint arXiv:1803.03807. Available at: https://arxiv.org/abs/1803.03807
64. Shi, L., Li, Y., Liu, T., Liu, J., Shan, B., Chen, H. (2019). Dynamic Distributed Honeypot Based on Blockchain. IEEE Access, 7, 54401–54410. https://doi.org/10.1109/ACCESS.2019.2920239
65. Zhuravchak, D. (2021). Ransomware spread prevention system using Python, auditd and Linux. Cybersecurity: Education, Science, Technique, 12(4), 108–116. https://doi.org/10.28925/2663-4023.2021.12.108116
66. Rudnichenko, Y., Melnyk, S., Havlovska, N., Illiashenko, O., Nakonechna, N. (2021). Strategic interaction of state institutions and enterprises with economic security positions in digital economy. WSEAS Transactions on Business and Economics, 18, 218–230. https://doi.org/10.37394/23207.2021.18.23
67. Hnylytska, L., Franchuk, V., Melnyk, S., Nakonechna, N., Leskiv, H., Hobela, V. (2022). Security-oriented model of business risk assessment. Financial and Credit Activity: Problems of Theory and Practice, 4 (45), 202–210. https://doi.org/10.55643/fcaptp.4.45.2022.3856
68. Jun, S., Szmajda, M., Khoma, V., Khoma, Y., Sabodashko, D., Kochan, O., Wang, J. (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.2020.132775

69. Khoma, V., Sabodashko, D., Kolchenko, V., Perepelytsia, P., Baranowski, M. (2024). Investigation of vulnerabilities in large language models using an automated testing system. CEUR Workshop Proceedings, 3826, 220–228. Available at: https://ceur-ws.org/Vol-3826/short10.pdf
70. Jia, J. (2023). Programmable system call security with eBPF. arXiv preprint arXiv:2302.10366. https://doi.org/10.48550/arXiv.2302.10366
71. Levin, J., Benson, T. A. (2020). ViperProbe: Rethinking microservice observability with eBPF. 2020 IEEE 9th International Conference on Cloud Networking (CloudNet), 1–8. https://doi.org/10.1109/CloudNet51028.2020.9335808
72. Edzuan Zainodin, M., et al. (2022). Entropy based method for malicious file detection. JOIV: International Journal on Informatics Visualization, 6 (4), 856–864. https://doi.org/10.30630/joiv.6.4.1265
73. Zhuravchak, D., Dudykevych, V. (2023). Challenges and prospects of implementing machine learning for real-time ransomware detection. Cybersecurity: Education, Science, Technique. Available at: https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/37567/127406.pdf
74. Liu, L., Wang, B., Yu, B., Zhong, Q., Pan, Y., Chen, X. (2017). Automatic malware classification and new malware detection using machine learning. Frontiers of Information Technology & Electronic Engineering, 18 (9), 1336–1347. https://doi.org/10.1631/fitee.1601325
75. Zhuravchak, D., Kiiko, E., Dudykevych, V. (2023). Using eBPF to identify ransomware that use DGA DNS queries. Information Technology and Security, 11 (2), 166–174. https://doi.org/10.20535/2411-1031.2023.11.2.293760
76. Alsaif, S. A. (2023). Machine learning-based ransomware classification of bitcoin transactions. Applied Computational Intelligence and Soft Computing, 2023, 1–10. https://doi.org/10.1155/2023/6274260
77. Kim, T., Ji, H., Im, E. G. (2018). Malware classification using machine learning and binary visualization. KIISE Transactions on Computing Practices, 24 (4), 198–203. https://doi.org/10.5626/ktcp.2018.24.4.198
78. Xuan, J., Jiang, H., Ren, Z., Zou, Q. (2018). Bayesian deep reinforcement learning via deep kernel learning. International Journal of Computational Intelligence Systems, 12 (1), 164–174. https://doi.org/10.2991/ijcis.2018.25905189
79. Mkandawire, Y., Zimba, A. (2023). A supervised machine learning ransomware host-based detection framework. Zambia ICT Journal, 7 (1), 52–56. https://doi.org/10.33260/zictjournal.v7i1.132
80. Fang, Z., Wang, J., Li, B., Wu, S., Zhou, H., Huang, Y. (2019). Evading anti-malware engines with deep reinforcement learning. IEEE Access, 7, 48867–48879. https://doi.org/10.1109/access.2019.2908033
81. Zhu, Y. (2023). Naive Bayesian spam filtering. Highlights in Science, Engineering and Technology, 38, 64–69. https://doi.org/10.54097/hset.v38i.5734
82. Liu, L., Wang, B., Yu, B., Zhong, Q., Pan, Y., Chen, X. (2017). Automatic malware classification and new malware detection using machine learning. Frontiers of Information Technology & Electronic Engineering, 18 (9), 1336–1347. https://doi.org/10.1631/fitee.1601325
83. Alhawi, O. M., Baldwin, J., Dehghantanha, A. (2018). Leveraging machine learning techniques for Windows ransomware network traffic detection. Digital Investigation, 24, 23–S31. https://doi.org/10.1016/j.diin.2018.01.007
84. Zhu, Y. (2023). Naive Bayesian spam filtering. Highlights in Science, Engineering and Technology, 38, 64–69. https://doi.org/10.54097/hset.v38i.5734

85. Zhang, K., Xu, H., Min, M. R. (2017). Collaborative support vector machine for malware detection. Procedia Computer Science, 108, 1682–1691. https://doi.org/10.1016/j.procs.2017.05.063
86. Zhuravchak, D. (2023). Ransomware monitoring with enhanced Berkeley Packet Filter (eBPF) and machine learning. Information Technology, Cybersecurity. https://doi.org/10.18372/2310-5461.60.18029
87. Thomas, T., Vijayaraghavan, A. P., Emmanuel, S. (2019). Support vector machines and malware detection. In Machine Learning Approaches in Cyber Security Analytics, 49–71. https://doi.org/10.1007/978-981-15-1706-8_4
88. Zhuravchak, D., Dudykevych, V., Tolkachova, A. (2023). Study of the structure of the system for detecting and preventing ransomware attacks based on endpoint detection and response. Cybersecurity: Education, Science, Technique, 3 (19), 69–82. https://doi.org/10.28925/2663-4023.2023.19.6982
89. Widagdo, G. B., Lim, C. (2017). Analysis of hybrid DDoS defense to mitigate DDoS impact. Advanced Science Letters, 23 (4), 3633–3639. https://doi.org/10.1166/asl.2017.9004
90. Fuloria, S. (2022). Cybersecurity and ransomware. Academia Letters. https://doi.org/10.20935/al4820
91. Zhuravchak, D., Dudykevych, V. (2023). Real-time ransomware detection by using eBPF and natural language processing and machine learning. IEEE Xplore. https://doi.org/10.1109/AICT61584.2023.10452697
92. Kret, T. (2024). Approaches to threat modeling in the creation of a comprehensive information security system for multi-level intelligent control systems. Computer Systems and Networks, 6 (1), 81–88. https://doi.org/10.23939/csn2024.01.081
93. 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. CEUR Workshop Proceedings, 3702, 400–412.
94. Yemanov, V., Dzyana, H., Dzyanyi, N., Dolinchenko, O., Didych, O. (2023). Modelling a public administration system for ensuring cybersecurity. International Journal of Safety and Security Engineering, 13 (1), 81–88. https://doi.org/10.18280/ijsse.130109
95. Ahmad, Z., et al. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. Transactions on Emerging Telecommunications Technologies, 32 (1), e4150. https://doi.org/10.1002/ett.4150
96. Tait, K.-A., et al. (2021). Intrusion Detection using Machine Learning Techniques: An Experimental Comparison. arXiv preprint arXiv:2105.13435. Available at: https://arxiv.org/abs/2105.13435
97. Gupta, C., Johri, I., Srinivasan, K., Hu, Y.-C., Qaisar, S. M., Huang, K.-Y. (2022). A Systematic Review on Machine Learning and Deep Learning Models for Electronic Information Security in Mobile Networks. Sensors, 22 (5), 2017. https://doi.org/10.3390/s22052017
98. Zhuravchak, D., Dudykevych, V., Tolkachova, A. (2023). Zero trust concept for Active Directory protection to detect ransomware. Cybersecurity: Education, Science, Technique, 2 (22), 179–190. https://doi.org/10.28925/2663-4023.2023.22.179190
99. Piskozub, A., Zhuravchak, D., Tolkachova, A. (2023). Researching vulnerabilities in chatbots with LLM (Large Language Model). Ukrainian Scientific Journal of Information Security, 29 (3), 166–172. https://doi.org/10.18372/2225-5036.29.18069

100. Zhuravchak, D., Opirskyy, I., Piskozub, A. (2022). Detection method of credential dumping through exploiting vulnerable Windows Error Reporting service in Windows operating systems. Modern Special Technics, 2 (69), 11–19. https://doi.org/10.36486/mst2411-3816.2022.2(69).2
101. Asmara, K., Fakhri, M., Raja, T. H. L. (2024). Analysis of Honeypot Networks and Intrusion Prevention System (IPS) on Wireless Networks. International Journal of Trend in Scientific Research and Development, 8 (1), 721–727. Available at: https://www.ijtsrd.com/papers/ijtsrd63502.pdf
102. Miano, S., Paolucci, F., Siracusa, D., Caviglione, L. (2023). A Highly Interactive Honeypot-Based Approach to Network Threat Analysis. Future Internet, 15 (4), 127. https://doi.org/10.3390/fi15040127
103. Wählisch, M., Vorbach, A., Keil, C., Schönfelder, J., Schmidt, T. C., Schiller, J. H. (2013). Design, Implementation, and Operation of a Mobile Honeypot. arXiv preprint arXiv:1301.7257. Available at: https://arxiv.org/abs/1301.7257
104. Jain, Y. K., Surabhi, S. (2011). Honeypot Based Secure Network System. International Journal on Computer Science and Engineering, 3 (2), 1003–1009. Available at: https://www.researchgate.net/publication/50247428_Honeypot_based_Secure_Network_System
105. Dedeoglu, V., Fischer, M. (2021). Blockchain-based Security Framework for IoT Devices in Smart Homes. Journal of Information Security and Applications, 58, 102748. https://doi.org/10.1016/j.jisa.2021.102748
106. Wazid, M., Hasan, R. (2019). A Blockchain-based secure and robust honeypot framework for smart cities. IEEE Access, 7, 101118–101131. https://doi.org/10.1109/ACCESS.2019.293062631.
107. Khan, M., Alqahtani, S. (2023). AI-driven threat detection in cloud computing: A survey on honeypot-based approaches. Future Generation Computer Systems, 137, 189–205. https://doi.org/10.1016/j.future.2023.04.015
108. 33. Sagirlar, G., Carminati, B., Ferrari, E. (2018). AutoBotCatcher: Blockchain-based P2P Botnet Detection for the Internet of Things. arXiv preprint arXiv:1809.10775. Available at: https://arxiv.org/abs/1809.10775
109. Commey, D., Hounsinou, S., Crosby, G. V. (2024). Strategic Deployment of Honeypots in Blockchain-based IoT Systems. arXiv preprint arXiv:2405.12951. Available at: https://arxiv.org/abs/2405.12951
110. Golomb, T., Mirsky, Y., Elovici, Y. (2018). CIoTA: Collaborative IoT Anomaly Detection via Blockchain. arXiv preprint arXiv:1803.03807. Available at: https://arxiv.org/abs/1803.03807
111. Partyka, O. (2024). Identifying attacks on the Bluetooth protocol using Wireshark and the Splunk SIEM system. In Technical Informatics and Artificial Intelligence: Engineer of XXI Century'2024. https://doi.org/10.53052/9788367652292.16
112. Dong, Y., Zampella, F., Alsehly, F. (2023). Beyond KNN: Deep Neighborhood Learning for WiFi-based Indoor Positioning Systems. arXiv preprint arXiv:2302.00810. https://doi.org/10.48550/arXiv.2302.00810
113. Belej, O., Nestor, N., Polotai, O. (2019). Developing a local positioning algorithm based on the identification of objects in a Wi-Fi Network of the Mall. International Conference on Perspective Technologies and Methods in MEMS Design, 32–36. https://doi.org/10.1109/MEMSTECH.2019.8817407
114. Kukharska, N., Lagun, A., Polotai, O. (2020). The steganographic approach to data protection using Arnold algorithm and the pixel-value differencing method. IEEE International Conference on Data Stream Mining and Processing, 174–177. https://doi.org/10.1109/DSMP47368.2020.9204205

115. Zhuravel, I., Semenyuk, S. (2024). Stochastic Models for Computer Malware Propagation. IEEE International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering, 424–427. https://doi.org/10.1109/TCSET64720.2024.10755827
116. Semenyuk, S. A., Chabanyuk, Y. M. (2024). Stochastic Evolutionary System With Markov-Modulated Poisson Perturbations in the Averaging Schema. Matematychni Studii, 62 (1), 102–108. https://doi.org/10.30970/ms.62.1.102-108
117. 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
118. 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
119. Tykholaz, D., Banakh, R., Mychuda, L., Piskozub, A., Kyrychok, R. (2024). Incident response with AWS detective controls. CEUR Workshop Proceedings, 3826, 190–197. Available at: https://ceur-ws.org/Vol-3826/short6.pdf
120. Volotovskyi, O., Banakh, R., Piskozub, A., Brzhevska, Z. (2024). Automated security assessment of Amazon Web Services accounts using CIS Benchmark and Python 3. CEUR Workshop Proceedings, 3826, 363–371. Available at: https://ceur-ws.org/Vol-3826/paper29.pdf
121. Lijuan Z. A Network Security Evaluation Method based on FUZZY and RST / Z. Lijuan, W. Qingxin // 2010 2nd International Conference on Education Technology and Computer (ICETC). 22-24 June 2010: proceedings. – Shanghai, China: IEEE, 2010, 40-44.
122. Runfu, Z., Lianfen, H., Mingbo, X. (2010). Security for Wireless Network Based on Fuzzy-AHP with Variable Weight. In 2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing, 2, 490–493. https://doi.org/10.1109/NSWCTC.2010.253
123. Ying-Chiang, C., Jen-Yi, P. (2014). Hybrid Network Defense Model Based on Fuzzy Evaluation. The Scientific World Journal, 2014, 1–12. https://doi.org/10.1155/2014/370865
124. Nyzhnyk, A., Partyka, A., Podpora, M. (2024). Increase the cybersecurity of SCADA and IIoT devices with secure memory management. CEUR Workshop Proceedings, 3800, 32–41. Available at: https://ceur-ws.org/Vol-3800/paper4.pdf
125. Goel, R., Sardana, A., Joshi, R. C. (2013). Wireless Honeypot: Framework, Architectures and Tools. International Journal of Network Security, 15 (5), 373–383. Available at: https://ijns.jalaxy.com.tw/contents/ijns-v15-n5/ijns-2013-v15-n5-p373-383.pdf
126. Banakh, R., Piskozub, A., Stefinko, Y. (2016). External elements of honeypot for wireless network. In Proceedings of the 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET 2016), 480–482. https://doi.org/10.1109/TCSET.2016.7452228
127. Ajah, I. A. (2014). Evaluation of Enhanced Security Solutions in 802.11-Based Networks. International Journal of Network Security & Its Applications (IJNSA), 6 (4), 29–42. https://doi.org/10.5121/ijnsa.2014.6403
128. Khanin, D., Otenko, V., Khoma, V. (2024). Research on the effectiveness of concatenated embeddings in facial verification. CEUR Workshop Proceedings, 3800, 12–21. https://ceur-ws.org/Vol-3800/paper1.pdf

129. Mychuda, Z., Mychuda, L., Zhuravel, I., Szcześniak, Z., Szcześniak, A. (2022). Modelling a new multifunctional high accuracy analogue-to-digital converter with an increased number of inputs. Electronics, 11 (11), Article 1677. https://doi.org/10.3390/electronics11111677
130. Mychuda, L., Myczuda, Z., Korobeinikova, T., Zhuravel, I., Romanyuk, O., Kotlyk, S. (2024). Optimization of Precision and Speed in ADCP. International Conference on Advanced Computer Information Technologies, 614–617. https://doi.org/10.1109/ACIT62333.2024.10712483
131. Vorobel, R. A., Zhuravel, I. M., Svirs'ka, L. M., Student, O. Z. (2011). Automatic selection and quantitative analysis of carbides on grain boundaries of 12Kh1MF steel after operation at a steam pipeline of a thermal power plant. Materials Science, 47 (3), 393–400. https://doi.org/10.1007/s11003-011-9408-3
132. Zhuravel, I. M., Michuda, L. Z. (2021). Application of the Mandelbrot–Zipf Law for the Quantitative Evaluation of the Average Size of Steel Grains. Materials Science, 57 (1), 80–85. https://doi.org/10.1007/s11003-021-00517-2
133. Khoma, V., Abibulaiev, A., Piskozub, A., Kret, T. (2024). Comprehensive Approach for Developing an Enterprise Cloud Infrastructure. CEUR Workshop Proceedings, 3654, 201–215. Available at: https://ceur-ws.org/Vol-3654/paper16.pdf
134. Forbes, G., Massie, S., Craw, S. (2020). WiFi-based human activity recognition using Raspberry Pi. In Proceedings of IEEE 32nd Tools with Artificial Intelligence International Conference (ICTAI 2020), 722–730. https://doi.org/10.1109/ICTAI50040.2020.00115
135. Lu, Q., Qu, H., Zhuang, Y., Lin, X. J., Ouyang, Y. (2018). Client-Side Evil Twin Attacks Detection Using Statistical Characteristics of 802.11 Data Frames. IEICE Transactions on Information and Systems, E101.D(10), 2465–2473. https://doi.org/10.1587/transinf.2018EDP7030
136. Modi, V., Parekh, A. (2017). Detection of Rogue Access Point to Prevent Evil Twin Attack in Wireless Network. International Journal of Engineering Research & Technology (IJERT), 6 (4), 69–74. https://doi.org/10.17577/IJERTV6IS040102
137. Opirskyy, I., Sovyn, Y., Mykhailova, O. (2022). Heuristic method of finding bitsliced-description of derivative cryptographic S-box. Proceedings of the 16th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET 2022), 104–109. https://doi.org/10.1109/TCSET55632.2022.9766933
138. Fedynyshyn, T., Opirskyy, I., Mykhaylova, O. (2023). A method to detect suspicious individuals through mobile device data. Proceedings of the 5th IEEE International Conference on Advanced Information and Communication Technologies (AICT 2023), 82–86. https://doi.org/10.1109/AICT58444.2023.10362549
139. 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. Available at: http://ceur-ws.org/Vol-3654/paper27.pdf
140. 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. Available at: https://ceur-ws.org/Vol-3550/paper5.pdf
141. 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), 6–13. https://doi.org/10.15587/1729-4061.2019.166349

142. Milov, O., Voitko, A., Husarova, I., Domaskin, O., Ivanchenko, Y., Ivanchenko, I., Korol, O., Kots, H., Opirskyy, I., Fraze-Frazenko, O. (2019). Development of methodology for modeling the interaction of antagonistic agents in cybersecurity systems. Eastern-European Journal of Enterprise Technologies, 2 (9) (98), 56–66. https://doi.org/10.15587/1729-4061.2019.164730
143. Yevseiev, S., Khokhlachova, Y., Ostapov, S., Laptiev, O., Korol, O., Milevskyi, S., Milov, O., Pohasii, S., Melenti, Y., Hrebeniuk, V., Havrylova, A. (2023). Models of Socio-Cyber-Physical Systems Security: Monograph. PC TECHNOLOGY CENTER. https://doi.org/10.15587/978-617-7319-72-5
144. Martseniuk, Y., Partyka, A., Harasymchuk, O., Shevchenko, S. (2024). Universal centralized secret data management for automated public cloud provisioning. CEUR Workshop Proceedings, 3826, 72–81. Available at: https://ceur-ws.org/Vol-3826/paper7.pdf
145. Martseniuk, Y., Partyka, A., Harasymchuk, O., Korshun, N. (2024). Automated Conformity Verification Concept for Cloud Security. CEUR Workshop Proceedings, 3654, 25–37. Available at: https://ceur-ws.org/Vol-3654/paper3.pdf
146. 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. Available at: https://ceur-ws.org/Vol-3550/short4.pdf
147. Opirskyy, I., Harasymchuk, O., Mykhaylova, O., Hrushkovskyi, O., Kozak, P. (2024). Pseudorandom sequence generator based on the computation of ln 2. CEUR Workshop Proceedings, 3829, 79–86. Available at: https://ceur-ws.org/Vol-3829/short10.pdf
148. Maksymovych, V., Mandrona, M., Harasymchuk, O. (2020). Dosimetric Detector Hardware Simulation Model Based on Modified Additive Fibonacci Generator. In Advances in Computer Science for Engineering and Education II, 162–171. https://doi.org/10.1007/978-3-030-16621-2_15
149. Harsha, S., Abdus Sattar, K., Sriramulu, B., Rao, V. (2019). Improving Wi-Fi security against evil twin attack using lightweight machine learning application. Compusoft, 8 (4), 3109–3115.
150. Kuo, E.-C., Chang, M.-S., Kao, D.-Y. (2018). User-side evil twin attack detection using time-delay statistics of TCP connection termination. 2018 20th International Conference on Advanced Communication Technology (ICACT), 211–216. https://doi.org/10.23919/ICACT.2018.8323699
151. Agarwal, M., Biswas, S., Nandi, S. (2018). An Efficient Scheme to Detect Evil Twin Rogue Access Point Attack in 802.11 Wi-Fi Networks. International Journal of Wireless Information Networks, 25 (2), 130–145. https://doi.org/10.1007/s10776-018-0396-1
152. Yang, C., Song, Y., Gu, G. (2012). Active User-Side Evil Twin Access Point Detection Using Statistical Techniques. IEEE Transactions on Information Forensics and Security, 7 (5), 1638–1651. https://doi.org/10.1109/TIFS.2012.2207383
153. Dong, Y., Zampella, F., Alsehly, F. (2023). Beyond KNN: Deep Neighborhood Learning for WiFi-based Indoor Positioning Systems. arXiv preprint arXiv:2302.00810. https://doi.org/10.48550/arXiv.2302.00810
154. Shanmugam, P., & Mandankandy, A. A. (2014). Study of honeypots: Analysis of WiFi honeypots and honeypots tools. Advances in Natural and Applied Sciences, 8(17), 48–59. https://www.researchgate.net/publication/341286903_Study_of_Honeypots_Analysis_of_WiFi_Honeypots_and_Honeypots_tools

155. Lu, Q., Qu, H., Zhuang, Y., Lin, X.-J., Ouyang, Y. (2018). Client-Side Evil Twin Attacks Detection Using Statistical Characteristics of 802.11 Data Frames. IEICE Transactions on Information and Systems, E101.D(9), 2465–2473. https://doi.org/10.1587/transinf.2018EDP7030
156. Hsu, F.-H., Wang, C.-S., Hsu, Y.-L., Cheng, Y.-P., Hsneh, Y.-H. (2016). A client-side detection mechanism for evil twins. Computers & Electrical Engineering, 59, 76–85. https://doi.org/10.1016/j.compeleceng.2015.10.010
157. Alotaibi, B., Elleithy, K. (2015). An empirical fingerprint framework to detect rogue access points. In 2015 IEEE Long Island Systems, Applications and Technology Conference (LISAT), 1–7. IEEE. https://doi.org/10.1109/LISAT.2015.7160194
158. Yu, J. (2014). Applying TCP profiling to detect wireless rogue access point. In Proceedings of the International Conference on Wireless Networks (ICWN 2014), 1–7. Available at: https://worldcomp-proceedings.com/proc/p2014/ICW3827.pdf
159. Petiz, I., Rocha, E., Salvador, P., Nogueira, A. (2013). Using multiscale traffic analysis to detect WPS attacks. In 2013 IEEE International Conference on Communications Workshops (ICC), 964–968. https://doi.org/10.1109/ICCW.2013.6649386
160. Holz, T., Raynal, F. (2005). Detecting honeypots and other suspicious environments. Proceedings of the Sixth Annual IEEE SMC Information Assurance Workshop, 29–36. https://doi.org/10.1109/IAW.2005.1495930
161. Srinivasa, S., Pedersen, J. M., Vasilomanolakis, E. (2023). Gotta catch 'em all: A multistage framework for honeypot fingerprinting. Digital Threats: Research and Practice, 4 (2), 12. https://doi.org/10.1145/3584976
162. Valeros, V., Rigaki, M., Garcia, S. (2023). Attacker profiling through analysis of attack patterns in geographically distributed honeypots. arXiv preprint arXiv:2305.01346. https://doi.org/10.48550/arXiv.2305.01346
163. García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. Computers & Security, 28 (1–2), 18–28. https://doi.org/10.1016/j.cose.2008.08.003
164. Oyeyemi, D. A., Ojo, A. K. (2024). SMS spam detection and classification to combat abuse in telephone networks using natural language processing. arXiv preprint arXiv:2406.06578. https://doi.org/10.48550/arXiv.2406.06578
165. Galán, F., Fernández, D. (2006). Use of VNUML in virtual honeynets deployment. In Proceedings of the 6th IEEE International Conference on Information Technology: Research and Education (ITRE 2006), 600–615. Available at: https://www.researchgate.net/publication/266094954_Use_of_VNUML_in_Virtual_Honeynets_Deployment
166. Agrawal, N., Tapaswi, S. (2017). The Performance Analysis of Honeypot Based Intrusion Detection System for Wireless Network. International Journal of Wireless Information Networks, 24 (1), 14–26. https://doi.org/10.1007/s10776-016-0330-3
167. Varadharajan, V., Tupakula, U. (2014). Security as a Service Model for Cloud Environment. IEEE Transactions on Network and Service Management, 11 (1), 60–75. https://doi.org/10.1109/TNSM.2014.041614.120394

168. Kondra, J. R., Bharti, S. K., Mishra, S. K., Babu, K. S. (2016, March). Honeypot-based Intrusion Detection System: A Performance Analysis. Paper presented at the 2016 3rd International Conference on Computing for Sustainable Global Development, Delhi, India. https://doi.org/10.13140/RG.2.1.4599.9768
169. Chatzoglou, E., Kampourakis, V., Kambourakis, G. (2023). BlOck: Paralyzing 802.11 connections through Block Ack frames. arXiv preprint arXiv:2302.05899. https://doi.org/10.48550/arXiv.2302.05899
170. AlQahtani, A. A. S., Alshayeb, T. (2023). Zero-Effort Two-Factor Authentication Using Wi-Fi Radio Wave Transmission and Machine Learning. arXiv preprint arXiv:2303.02503. https://doi.org/10.48550/arXiv.2303.02503
171. Manev, A. (2023). Tamper-Evident Pairing. arXiv preprint arXiv:2311.14790. https://doi.org/10.48550/arXiv.2311.14790
172. Vanhoef, M. (2021). Fragment and Forge: Breaking Wi-Fi Through Frame Aggregation and Fragmentation. In Proceedings of the 30th USENIX Security Symposium, 19–36. Available at: https://www.usenix.org/conference/usenixsecurity21/presentation/vanhoef
173. Banakh, R., Piskozub, A. (2018). Attackers' Wi-Fi devices metadata interception for their location identification. In Proceedings of the 2018 IEEE 4th International Symposium on Wireless Systems within the International Conferences on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS-SWS 2018), 112–116. https://doi.org/10.1109/IDAACS-SWS.2018.852553870
174. Barthe, G., Cauligi, S., Grégoire, B., Koutsos, A., Liao, K., Oliveira, T., Zanella-Béguelin, S. (2021). High-Assurance Cryptography in the Spectre Era. In Proceedings of the 2021 IEEE Symposium on Security and Privacy (SP), 1884–1901. https://doi.org/10.1109/SP40001.2021.00101
175. Benger, N., van de Pol, J., Smart, N. P., Yarom, Y. (2014). "Ooh Aah... Just a Little Bit": A small amount of side channel can go a long way. In Cryptographic Hardware and Embedded Systems (CHES 2014), LNCS, vol. 8731, 75–92. https://doi.org/10.1007/978-3-662-44709-3_5
176. Bernstein, D. J., Breitner, J., Genkin, D., van Groot Bruinderink, L., Heninger, N., Lange, T., Yarom, Y. (2017). Sliding Right into Disaster: Left-to-Right Sliding Windows Leak. In Cryptographic Hardware and Embedded Systems (CHES 2017), LNCS, vol. 10529, 555–576. https://doi.org/10.1007/978-3-319-66787-4_27
177. Zhang, R., Huang, L., Xiao, M. (2010). Security evaluation for wireless network based on fuzzy-AHP with variable weight. In Proceedings of the 2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing, 2, 490–493. https://doi.org/10.1109/NSWCTC.2010.122
178. Ajah, I. A. (2014). Evaluation of enhanced security solutions in 802.11-based networks. International Journal of Network Security & Its Applications (IJNSA), 6 (4), 29–42. https://doi.org/10.5121/ijnsa.2014.6403
179. Mohammad Ali Pour, F., Rashidi, M. (2024). From WEP to WPA3, Red Teamer's Guide to Wi-Fi Exploits. Zenodo. https://doi.org/10.5281/zenodo.14039895
180. Cahyadi, D., Astuti, I. F. (2021). Comparison of throughput and CPU usage between WPA3 and WPA2 security methods on wireless networks 802.11n. AIP Conference Proceedings, 2482 (1), 040006. https://doi.org/10.1063/5.0110514
181. Faíscas, D. (2022). (In)Security in Wi-Fi networks: A systematic review. ARIS2 – Advanced Research on Information Systems Security. https://doi.org/10.56394/aris2.v2i2.18

182. Schepers, D., Ranganathan, A., Vanhoef, M. (2022). On the robustness of Wi-Fi deauthentication countermeasures. In Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '22), 245–256. https://doi.org/10.1145/3507657.3528548
183. Felter, W., Ferreira, A., Rajamony, R., Rubio, J. (2015). An updated performance comparison of virtual machines and Linux containers. In Proceedings of the 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 171–172. IEEE. https://doi.org/10.1109/ISPASS.2015.7095802
184. Guan, C., Fu, X. (2023). HoneyIoT: Adaptive High-Interaction Honeypot for IoT Devices. Proceedings of the 2023 ACM SIGSAC Conference on Security and Privacy in Wireless and Mobile Networks (WiSec '23). Available at: https://mcn.cse.psu.edu/paper/guan-chongqi/wisec23-chongqi.pdf
185. Soundararajan, R., Rajagopal, M., Muthuramalingam, A., Hossain, E., Lloret, J. (2022). Interleaved Honeypot-Framing Model with Secure MAC Policies for Wireless Sensor Networks. Sensors, 22 (20), 8046. https://doi.org/10.3390/s22208046
186. De Almeida Braga, D., Kulatova, N., Sabt, M., Fouque, P.-A., Bhargavan, K. (2023). From Dragondoom to Dragonstar: Side-channel Attacks and Formally Verified Implementation of WPA3 Dragonfly Handshake. arXiv preprint arXiv:2307.09243. https://doi.org/10.48550/arXiv.2307.09243
187. Stefinko, Y., Piskozub, A., Banakh, R. (2016). Manual and Automated Penetration Testing: Benefits and Drawbacks. Modern Tendency. In Proceedings of the XIIIth International Conference on Modern Problems of Radio Engineering, Telecommunications, and Computer Science (TCSET'2016), 488–491. https://doi.org/10.1109/TCSET.2016.7452230
188. Chatzoglou, E., Kambourakis, G., Kolias, C. (2022). How is your Wi-Fi connection today? DoS attacks on WPA3-SAE. Journal of Information Security and Applications, 64, 103060. https://doi.org/10.1016/j.jisa.2021.103060
189. Lu, H.-J., Yu, Y. (2021). Research on WiFi penetration testing with Kali Linux. Complexity, 2021, 1–8. https://doi.org/10.1155/2021/5570001
190. Ahmad, N. (2017). Cloud computing: Technology, security issues and solutions. In Proceedings of the 2nd International Conference on Anti-Cyber Crimes (ICACC 2017), 30–35. https://doi.org/10.1109/Anti-Cybercrime.2017.7905254
191. Tissir, N., El Kafhali, S., Aboutabit, N. (2020). Cloud computing security classifications and taxonomies: A comprehensive study and comparison. In Proceedings of the 2020 International Conference on Cloud Computing Technologies and Applications (CloudTech), 1–6. IEEE. https://doi.org/10.1109/CloudTech49835.2020.9365884
192. Banakh, R., Piskozub, A., Opirskyy, I. (2019). Detection of MAC spoofing attacks in IEEE 802.11 networks using signal strength from attackers' devices. In Advances in Intelligent Systems and Computing, 754, 468–477). https://doi.org/10.1007/978-3-319-91008-6_47
193. Mladenova, T., Valova, I. (2021). Analysis of the KNN classifier distance metrics for Bulgarian fake news detection. In Proceedings of the 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA 2021), 1–4. IEEE. https://doi.org/10.1109/HORA52670.2021.946133

194. Taunk, K., De, S., Verma, S., Swetapadma, A. (2019). A brief review of nearest neighbor algorithm for learning and classification. In Proceedings of the 2019 International Conference on Intelligent Computing and Control Systems (ICCS), 1255–1260. IEEE. https://doi.org/10.1109/ICCS45141.2019.9065747
195. Arias, J., Budde, C. E., Jansen, N. (2020). Hackers vs. Security: Attack-Defence Trees as Asynchronous Multi-agent Systems. In Formal Methods and Software Engineering, 3–19. https://doi.org/10.1007/978-3-030-63406-3_1
196. Vanhoef, M., Ronen, E. (2020). Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd. 2020 IEEE Symposium on Security and Privacy (SP), 517–533. https://doi.org/10.1109/SP40000.2020.00031
197. Nagpal, J., Patil, R., Jain, V., Pokhriyal, R., Rajawat, R. (2018). Evil Twin Attack and Its Detection. International Journal of Emerging Technologies and Innovative Research, 5 (12), 169–171. Retrieved from https://www.jetir.org/papers/JETIR1812B26.pdf
198. Bednarczyk, M., Piotrowski, Z. (2019). Will WPA3 really provide Wi-Fi security at a higher level? In Proceedings of SPIE 11055, XII Conference on Reconnaissance and Electronic Warfare Systems, 1105514. https://doi.org/10.1117/12.2525020
199. Banakh, R., Piskozub, A., Opirskyy, I. (2023). Devising a method for detecting "Evil Twin" attacks on IEEE 802.11 networks (Wi-Fi) with KNN classification model. Eastern-European Journal of Enterprise Technologies, 3 (9 (123)), 20–32. https://doi.org/10.15587/1729-4061.2023.280693
200. Salkind, N. J., Frey, B. B. (2019). Statistics for people who (think they) hate statistics. SAGE Publications, Inc. https://doi.org/10.4135/9781071802412

Edited by
Oleh Harasymchuk

INTELLIGENT CYBER DEFENCE SYSTEMS: DETECTION OF RANSOMWARE AND PROTECTION OF WIRELESS NETWORKS BASED ON ARTIFICIAL INTELLIGENCE TECHNOLOGIES

Ivan Opirskyy, Roman Banakh, Danyil Zhuravchak, Oleh Harasymchuk, Olha Partyka, Andrian Piskozub, Elena Nyemkova, Sviatoslav Vasylyshyn, Andrii Partyka, Yuriy Nakonechnyy, Taras Lukovskyy, Vitalii Susukailo, Viktor Otenko, Ivan Tyshyk, Nazarii Dzianyi, Dmytro Sabodashko, Petro Haraniuk, Valerii Dudykevych, Serhiy Semenyuk, Marta Stakhiv, Ihor Zhuravel, Taras Kret, Lesya Mychuda, Zynoviy Mychuda, Orest Polotai, Yevhenii Kurii, Nataliya Nakonechna, Nataliya Luzhetska, Anatoliy Obshta, Tetiana Korobeinikova

Monograph

Technical editor I. Prudius
Desktop publishing T. Serhiienko

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