Ministry of Science and Higher Education of the Russian Federation
Federal State Autonomous Educational Institution of Higher Education
«Ural Federal University named after the first President of Russia B.N. Yeltsin»

Engineering School of Information Technologies,
Telecommunications and Control Systems

School of Professional and Academic Education

ADMIT TO THESIS DEFENSE BEFORE THE SEC

Head of EP 09.04.03 M.A. Medvedeva

____

«01» June 2024

MASTER THESIS

Development of methods and algorithms for intrusion detection and prevention systems based on statistical methods and sustainable machine learning algorithms

Research supervisor: Associate Professor, Candidate of Economic Sciences	________________ signature	Medvedev M.A.
Research supervisor: Senior lecturer,	________________ signature	Agbozo E.
Student: Group number RIM-210980	________________ signature	Kirin E.D.

Yekaterinburg

2024

ABSTRACT

Topic of master's thesis:

Development of methods and algorithms for intrusion detection and prevention systems based on statistical methods and sustainable machine learning algorithms

The master thesis has been written on 114 pages and contains 14 tables, 56 figures, 42 references.

Researching vulnerabilities in intrusion detection systems (IDS/IPS) using algorithms based on statistical methods and machine learning is a pertinent topic due to the continuous rise of cyber threats, the necessity of data privacy protection, the application of cutting-edge technologies, and the widespread use of machine learning methods in the field of information security.

The practical significance of this research lies in the following aspects: the findings will help identify vulnerabilities in intrusion detection systems (IDS/IPS), thus enhancing the overall security level of information systems; studying algorithms based on statistical methods and machine learning will facilitate the development of new methods for defense against attacks and their integration into existing IDS/IPS systems; the obtained results can be utilized for training information security specialists, thereby enhancing qualification levels and preparing personnel in this field.

The economic efficiency of the research directions can be assessed as follows: the use of improved protection algorithms will mitigate cyber attack risks, data breaches, and other incidents, consequently reducing organizational losses associated with security breaches; ensuring reliable protection of information systems from external threats enhances organizational reputation, increases customer and partner trust, potentially leading to expanded business activities and attracting new clients.

The scientific novelty proposed by this research involves refining existing vulnerability detection algorithms in IDS/IPS systems, based on a combination of statistical methods and machine learning techniques. Additionally, the feasibility of

applying machine learning methods to detect hidden and advanced attacks, which traditional IDS/IPS systems may overlook, will be explored.

The research will utilize a wide range of data, including real cyber attack data for experimental testing. Previous research findings in cybersecurity and intrusion detection systems will also be examined.

The object of the research is intrusion detection systems (IDS/IPS) utilizing algorithms based on statistical methods and machine learning techniques. The focus of the study is on the system itself, its components, detection algorithms, and mechanisms, as well as its operational principles in the context of identifying vulnerabilities and potential attacks.

The subject of the research includes intrusion detection algorithms in IDS/IPS based on statistical methods and machine learning.

Based on existing research and literary sources, the research objectives and goals have been identified. The main goal is to evaluate IDS/IPS based on statistical methods and machine learning. The project tasks include:

1. Studying intrusion detection algorithms based on statistical methods and machine learning.
2. Analyzing vulnerabilities in IDS/IPS systems and identifying common types of attacks.
3. Studying methods to protect IDS/IPS from common attacks.
4. Developing a test environment to research vulnerabilities in intrusion detection systems.
5. Evaluating the effectiveness of intrusion detection algorithms.
6. Comparing intrusion detection algorithms based on statistical methods and machine learning.
7. Identifying the most effective intrusion detection algorithm based on statistical methods and machine learning.

CONTENTS

INTRODUCTION

Modern information technologies play a crucial role in ensuring the national security of the Russian Federation. Presidential Decree of the Russian Federation dated May 1, 2022, No. 250 "On Approval of the Strategy for the Development of the Information Society in the Russian Federation for 2022-2030" identifies one of the main directions for the development of the information society in Russia as ensuring the security of critical information infrastructure. In this context, researching vulnerabilities in intrusion detection systems (IDS/IPS), which are important elements of information system protection, becomes particularly relevant.

Intrusion detection systems are designed to detect and prevent unauthorized access to information resources. However, despite their importance, these systems may contain various vulnerabilities that malicious actors can exploit to penetrate protected information systems. Researching such vulnerabilities is a necessary condition for increasing the effectiveness of protecting the critical information infrastructure of the Russian Federation.

Studying vulnerabilities in intrusion detection systems (IDS/IPS) using algorithms based on statistical methods and machine learning is a topical issue due to the constant growth of cyber threats, the need to protect data confidentiality, the application of cutting-edge technologies, and the prevalence of machine learning methods in the field of information security.

Conducting research to study vulnerabilities in IDS allows for identifying and addressing weaknesses in intrusion detection systems, thereby improving their effectiveness and accuracy in detecting attacks. This is particularly important in the face of constantly evolving cyber threats, as malicious actors continuously develop new methods and techniques for intrusion.

Research on vulnerabilities in IDS also contributes to enhancing overall information security, as it helps detect and mitigate vulnerabilities that could be exploited to bypass defense systems and conduct successful cyber-attacks.

Works by authors Le Kuang Min, Nguyen An Chuen, Nguyen Chung Thien, and Fan Hue Anh address the challenges of improving the effectiveness of traditional methods used for detecting network anomalies in network systems. Substantial contributions to studying this issue were made by Glushchenko M.V., Glushchenko S.A., Shiryayev A.A., who examined the types and methods of intrusion detection systems in the information structure of enterprises. Their work reflects the nature of intrusion detection systems based on anomaly methods, which typically search for network traffic deviating from the standard network behavior model. Alshaibi A.D., Al-Ani M.M., Konev A.A. had a significant impact on solving the problem of this research. Their works contain fundamental principles of machine learning models, advantages, and limitations of all detection methods, providing the basis for developing Intrusion Detection Systems (IDS).

The practical significance of the research lies in the following aspects: the research results will help identify vulnerabilities in intrusion detection systems (IDS/IPS), thereby enhancing the overall level of information system security; studying algorithms based on statistical methods and machine learning will enable the development of new methods for protection against attacks and their implementation in existing IDS/IPS systems; the obtained results can be used for training information security specialists, contributing to raising the qualification level and preparing personnel in this field.

Modern technologies and internet infrastructure permeate all aspects of our lives, from banking operations to critical infrastructure systems. With the increasing number of devices connected to the internet and the volume of digital data, the risk of cyber-attacks also rises. IDS/IPS play a crucial role in detecting and preventing such attacks.

Hackers continuously refine their methods to bypass existing defense systems. Traditional intrusion detection approaches based on statistical methods and rules are becoming less effective against new and advanced threats.

Machine learning (ML) is a powerful tool in cybersecurity, enabling threat detection through the analysis of large volumes of data and the identification of hidden patterns. However, there is a risk that attackers may also use ML to create more sophisticated and stealthy attacks that evade existing IDS/IPS systems.

Given these factors, there is an evident need to improve intrusion detection systems. Researching vulnerabilities in existing IDS/IPS systems based on both statistical methods and machine learning will help identify and address their weaknesses, enhancing the effectiveness of detecting and preventing cyber-attacks.

The results of this research can be directly applied to enhancing the security of information systems in both corporate and government sectors, as well as in developing new methods for detecting and preventing cyber-attacks.

Thus, researching vulnerabilities in intrusion detection systems (IDS/IPS) using algorithms based on statistical methods and machine learning is a relevant and important task in the field of cybersecurity, which requires further research and development.

To achieve the research goals, a comprehensive approach will be used, including literature analysis and experimental studies. Specifically, data analysis, examination of machine learning algorithms, and testing on real datasets and attack simulations will be conducted.

The scientific novelty offered by this research includes refining existing algorithms for detecting vulnerabilities in IDS/IPS systems based on a combination of statistical methods and machine learning. Additionally, the potential application of machine learning methods for detecting hidden and advanced attacks that traditional IDS/IPS systems may overlook will be explored.

The research will focus on intrusion detection systems (IDS/IPS) that use algorithms based on statistical methods and machine learning. The study will concentrate on the system itself, its components, detection algorithms, and mechanisms, as well as its operation principles in the context of identifying vulnerabilities and possible attacks.

The subject of the research is intrusion detection algorithms in IDS/IPS systems based on statistical methods and machine learning.

Based on existing research and literature sources, the research goals and objectives have been identified. The main goal is to evaluate IDS/IPS based on statistical methods and machine learning. Project tasks include:

1. Studying intrusion detection algorithms based on statistical methods and machine learning.
2. Analyzing vulnerabilities in IDS/IPS systems and identifying common attacks.
3. Studying methods to protect IDS/IPS from common attacks.
4. Developing a test environment for investigating vulnerabilities in intrusion detection systems.
5. Evaluating the effectiveness of intrusion detection algorithms.
6. Comparing intrusion detection algorithms based on statistical methods and machine learning.
7. Identifying the most effective intrusion detection algorithm based on statistical methods and machine learning.

1 INTRUSION DETECTION SYSTEMS (IDS/IPS) THAT UTILIZE ALGORITHMS BASED ON STATISTICAL METHODS AND MACHINE LEARNING.

1.1 Justification of the relevance of research on vulnerabilities in intrusion detection systems (IDS/IPS).

The relevance of a systematic literature review on vulnerabilities in intrusion detection systems (IDS/IPS) and their widespread use for protecting various organizational networks has been demonstrated by several scholarly works on this topic.

Glushchenko M.V., Shiryaev A.A., Glushenko S.A. [1] investigated the types and methods of intrusion detection systems in the information structure of enterprises. They found that intrusion detection systems based on anomaly detection methods typically search for network traffic that differs from the standard behavior model of the network. The main principle is that network traffic behavior during an attack significantly differs from normal user traffic. IDS using anomaly detection methods create a profile (model of normal network traffic behavior) based on the standard behavior of network traffic in the network. When such IDS detects differences in the current network traffic behavior from the saved profile, an intrusion is recorded. Intrusion detection systems based on anomaly detection methods are capable of detecting new attacks whose signatures have not yet been identified. However, it was also found that the main drawback of this method is false positives when network traffic behavior deviates from the created profile.

Le Quang Minh, Fan Huy Anh, Nguyen Anh Chuen, Nguyen Chung Thien [2] concluded that traditional methods used in modern network systems to detect network anomalies are becoming outdated and ineffective in the face of changing hacker attacks and methods. In this study, the authors presented an intelligent model of an IPS/IDS system that combines machine learning with the development of additional updates before new network attacks to improve IDS transmission systems. Thus, this helps the system effectively prevent attacks even with new types of hacker attacks. The authors' team created an IDS system based on machine learning, self-learning, and intelligent reasoning, where new attacks are based on collected datasets. However, the proposed system still has some shortcomings, such as: the dataset was built over a long period, there are not many updates, new forms of attack have not been added, and there is a lack of accuracy.

Kumaga N.K., Grigoryevych A.V. [3] studied the design and implementation of an intrusion detection and prevention system "IDS/IPS" in the corporate network of UGTU. They found that currently, in UGTU, to ensure security or protect information, intermediate access tools (Proxy Server), firewalls (Firewall), and antivirus protection tools are used. Using only these information security mechanisms does not fully and effectively detect and prevent unauthorized and malicious activity in the UGTU network. The authors concluded that the following problems arise from this:

1. Unauthorized access to the network and systems.
2. Unauthorized use of IP telephony.
3. Hacking of sites and web applications.
4. Encryption of users' computers for ransom.

To address these issues, the authors propose using network resources from external attacks and supplementing existing technologies, which will allow timely detection and prevention of IDS/IPS intrusions.

A.D. Alshaibi, M.M. Al-Ani, A.A. Konev [4] conducted an extensive systematic literature review. They analyzed machine learning models and provided information about the advantages and limitations of all detection methods, laying the groundwork for the development of Intrusion Detection Systems (IDS). Machine learning methods (ANN, SVM, KBS) are widely used for developing IDS to timely and automatically detect and classify cyberattacks. This study provides a general overview of various approaches to creating machine learning algorithms, their major pros and cons, and helps select the appropriate algorithm based on the dimensionality and type of input data.

Currently, we observe dynamic growth in scientific activity concerning the vulnerability assessment of IDS/IPS. Bazhenov I.O. [5] showed that applying ready-made intelligent attack detection tools "out of the box" to the anomaly detection task leads to a high number of false positives and misses attacks because network traffic is a stream that changes daily. Therefore, one approach to solving the attack detection problem is dynamic, adaptive adjustment of intelligent detectors.

However, in the vast majority of attack detection systems, rule-based methods are primarily used, as the created rules provide justification for recording an attack at a specific moment and allow for easier system configuration, whereas intelligent attack detection methods, due to their complex learning algorithm, represent a black box.

Some authors [6] examined methods and means of ensuring information security in a local computer network and implementing a module of an intrusion detection and prevention system based on it, presented in the form of a deceptive system, which allows combating various network threats by setting traps and falsifying system parameters. The flexibility of configuration is a feature of such a solution.

Some authors [7] consider signature analysis as an effective preventive measure against intrusions. However, they conclude that the problem of detecting cybercrimes is complicated by gaps in existing legislation and that IDS/IPS systems require continuous improvement and modernization.

Algorithms based on statistical methods and machine learning techniques significantly enhance the process of detecting new network attacks, learning to identify them correctly, block them, and prevent future threats.

Analysis of related research in this area has shown an increasing number of high-quality articles identifying practical and theoretical issues in using IDS/IPS systems. However, most studies still emphasize the need for continued in-depth research and systematic study of vulnerabilities in intrusion detection systems (IDS/IPS).

The bibliometric indicators of the corresponding research direction are presented in Figures 1-3. The trends indicate a growing interest of the scientific community in research in this direction.

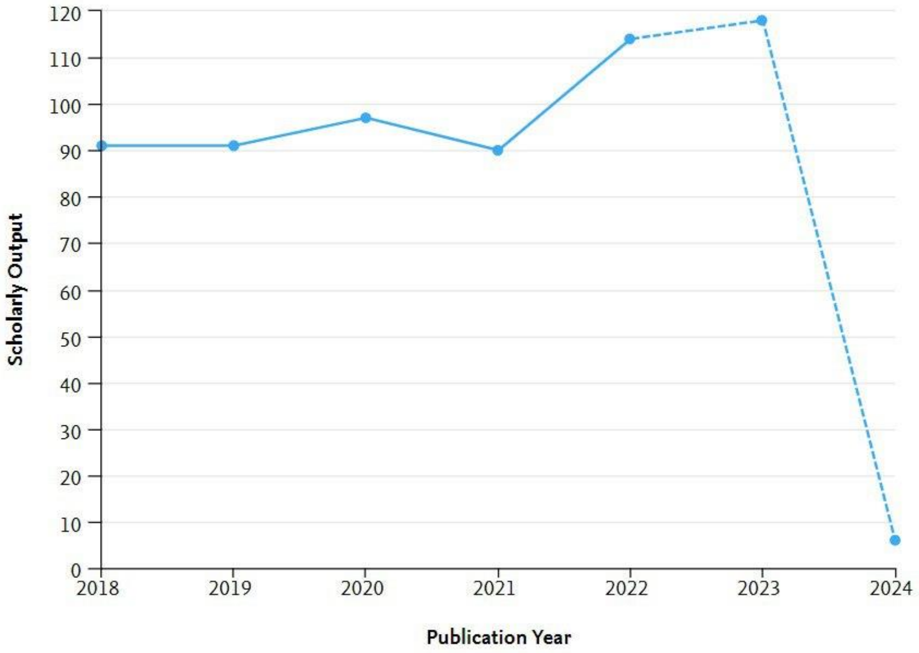


Figure 1 - Scientometric Indicators

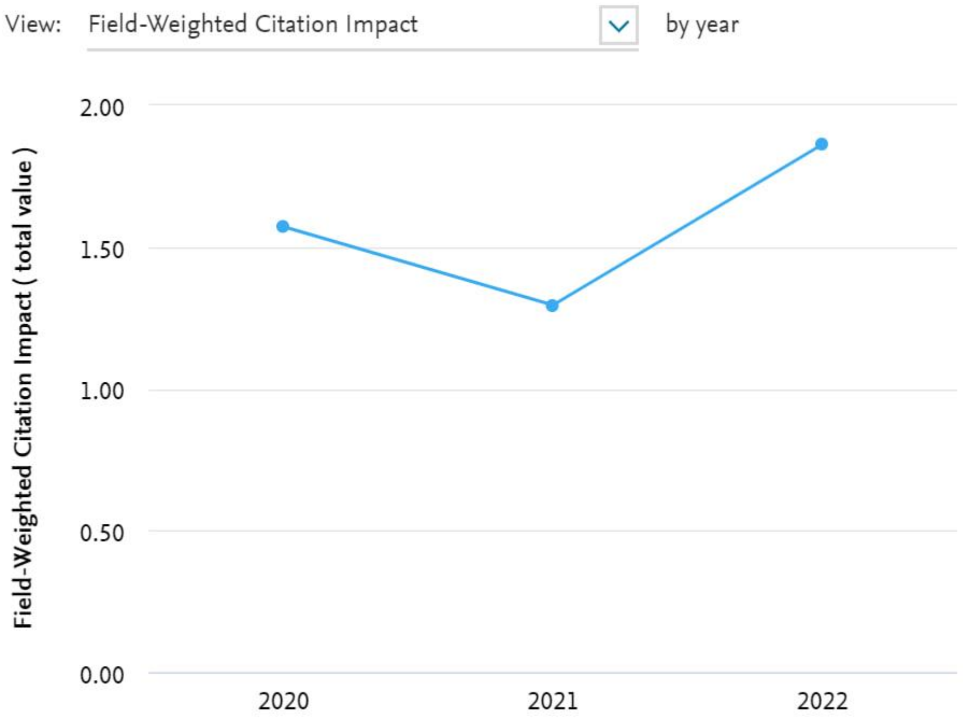

View: Field-Weighted Citation Impact by year

Figure 2 - Scientometric Indicators

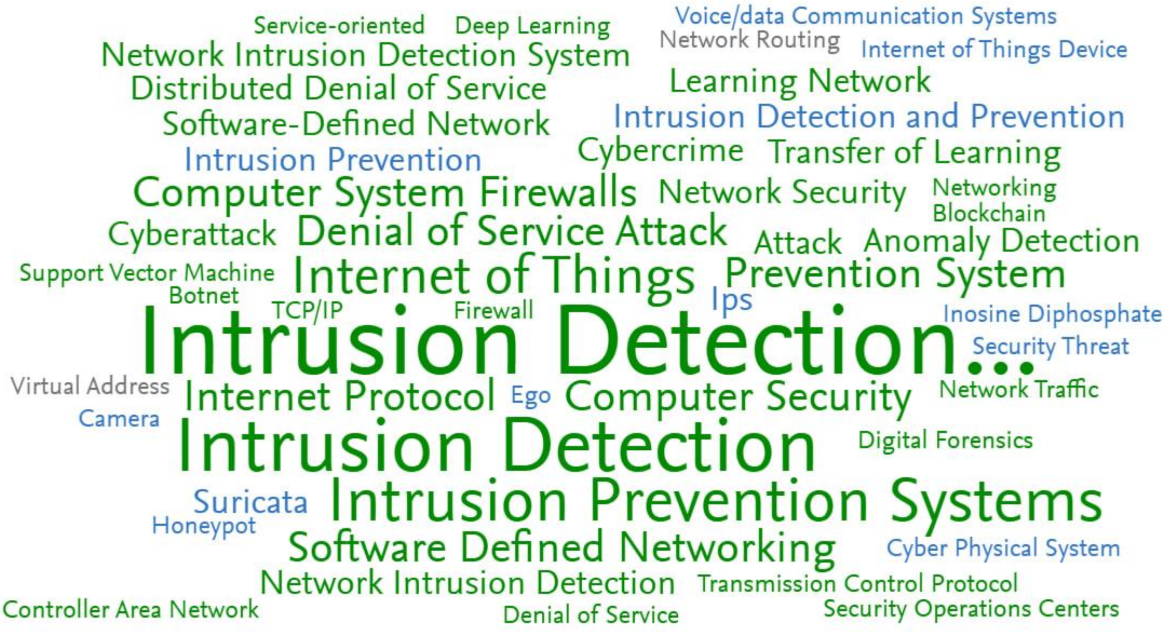


Figure 3 - Scientometric Indicators

1.2 Extraction of Data on IDS/IPS Methods Based on Statistical Methods and Machine Learning Techniques

Inclusion Criteria for the Review:

- Original articles and conference papers describing the study of IDS/IPS vulnerabilities based on statistical methods and machine learning techniques, research on intrusion detection algorithms, and evaluation of algorithm effectiveness against various typical attacks.
- Patent documentation containing descriptions of statistical methods and machine learning techniques applied in IDS/IPS systems, along with detailed descriptions of IDS/IPS system architectures.

Exclusion Criteria:

- Documents and conference materials on algorithms not based on statistical methods and machine learning techniques;

- Articles and materials inaccessible via Ural Federal University's corporate subscription.

Research Question for the Literature Review: What are the existing methods of intrusion detection systems (IDS/IPS) based on statistical methods and machine learning techniques? For example, XGBoost, KNN, SVM, etc.?

Potential Users of the Results: specialists in cybersecurity and organizational system protection, researchers of intrusion detection systems,

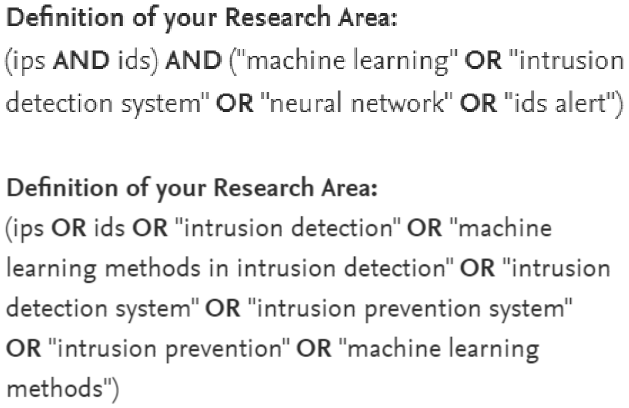
Analysis and evaluation of intrusion detection system (IDS/IPS) algorithms based on statistical methods and machine learning techniques.

Practical Outcome - evaluation of the effectiveness of IDS/IPS protection methods based on experiments, reducing detection vulnerabilities in systems with algorithms based on statistical methods and machine learning techniques.

The literature review can be applied in both industrial (practical) and scientific (research) environments.

Description of the Search Process:

1. Selected Libraries: Elsevier, eLibrary.
2. Selected Timeframe: 2017-2023.
3. Quality Criteria: Only articles indexed in RSCI, HAC, and Scopus.
4. Examples of Search Queries:



Definition of your Research Area:
(ips **AND** ids) **AND** ("machine learning" **OR** "intrusion detection system" **OR** "neural network" **OR** "ids alert")

Definition of your Research Area:
(ips **OR** ids **OR** "intrusion detection" **OR** "machine learning methods in intrusion detection" **OR** "intrusion detection system" **OR** "intrusion prevention system" **OR** "intrusion prevention" **OR** "machine learning methods")

The step-by-step process for analyzing the documentation is presented in Table 1.

Table 1 - Publication Analysis Process

	Elibrary	Elsevier
Step 0 - Query	Found: 67	Found: 729
Step 1 - Full-text Availability Check	Available in full text: 23	Available in full text: 287
Step 2 - Title and Metadata Analysis	Remaining: 11	Remaining: 184
Step 3 - Abstract Analysis	Remaining: 7	Remaining: 37
Step 4 - Result Analysis	Remaining: 4	Remaining: 10

The main goal of the research is to extract methods of intrusion detection systems (IDS/IPS) based on statistical methods and machine learning techniques from the documents.

From the IDS/IPS algorithms, it is necessary to extract models based on statistical methods and those related to machine learning.

The primary aim of data synthesis is to compare existing statistical methods and machine learning techniques to assess the vulnerabilities of IDS/IPS and to identify recommendations for addressing these vulnerabilities in IDS/IPS algorithms.

For the search, the technical field was determined by selecting IPC indexes:

1. **G06F 21/57 -** Certification or maintaining trusted computer platforms, such as secure booting or shutting down, version control, software system checks, secure updates, or vulnerability assessment [2013.01].
2. **G06F 21/55 -** Local intrusion detection or countermeasures [2013.01].
3. **G06N 20/00 -** Machine learning [2019.01].
4. **G06F 21/00 -** Devices for protecting computers, their components, programs, or data against unauthorized activity [2013.01].

For the analysis of patent documentation, the following databases were used: Rospatent, Google Patents.

We analyzed the documentation of patented tools, methods, and algorithms in the field of information system and data protection, the documentation of patented methods and algorithms for machine learning applied in the field of information security, and the documentation of registered software models for electronic computers (EC).

The main goal is to extract from the patent documentation the machine learning methods and algorithms used in the field of information security, as well as the methods and algorithms for the functioning of intrusion detection systems (IDS/IPS) based on statistical methods and machine learning techniques.

Patent Search Queries for Rospatent:

1. Main Query Area: Intrusion Detection System OR Intrusion Prevention System

Total Found: 141

Selected Search Bases (number of documents found):

Abstracts of Russian Inventions (RI): 25

Applications for Russian Inventions (ZIZ): 46

Full Texts of Russian Inventions from the Last Three Bulletins (NIZ): 45

Formulas of Russian Utility Models (FPM): 11

Formulas of Russian Utility Models from the Last Three Bulletins (NPM): 3

Prospective Russian Inventions (PI): 11

2. Main Query Area: Intrusion Prevention System OR Intrusion detection system

Total Found: 13

Selected Search Bases (number of documents found):

Abstracts of Russian Inventions (RI): 0

Applications for Russian Inventions (ZIZ): 0

Full Texts of Russian Inventions from the Last Three Bulletins (NIZ): 10

Formulas of Russian Utility Models (FPM): 0

Formulas of Russian Utility Models from the Last Three Bulletins (NPM): 0

Prospective Russian Inventions (PI): 3

3. Main Query Area: (Intrusion Prevention System OR Intrusion detection system OR система обнаружения вторжений) AND машинное обучение

Total Found: 7

Selected Search Bases (number of documents found):

Abstracts of Russian Inventions (RI): 0

Applications for Russian Inventions (ZIZ): 3

Full Texts of Russian Inventions from the Last Three Bulletins (NIZ): 3

Formulas of Russian Utility Models (FPM): 1

Formulas of Russian Utility Models from the Last Three Bulletins (NPM): 0

Prospective Russian Inventions (PI): 0

4. Main Query Area: (Intrusion Prevention System OR Intrusion detection system OR система обнаружения вторжений) AND (машинное обучение OR статистические методы)

Total Found: 12

Selected Search Bases (number of documents found):

Abstracts of Russian Inventions (RI): 1

Applications for Russian Inventions (ZIZ): 3

Full Texts of Russian Inventions from the Last Three Bulletins (NIZ): 6

Formulas of Russian Utility Models (FPM): 1

Formulas of Russian Utility Models from the Last Three Bulletins (NPM): 0

Prospective Russian Inventions (PI): 1

5. Main Query Area: Система обнаружения вторжений OR Система предотвращения вторжений AND (машинное обучение OR статистические методы)

Total Found: 123

Selected Search Bases (number of documents found):

Abstracts of Russian Inventions (RI): 21

Applications for Russian Inventions (ZIZ): 38

Full Texts of Russian Inventions from the Last Three Bulletins (NIZ): 39

Formulas of Russian Utility Models (FPM): 11

Formulas of Russian Utility Models from the Last Three Bulletins (NPM): 3

Prospective Russian Inventions (PI): 11

6. Main Query Area: машинное обучение and атаки

Total Found: 27

Selected Search Bases (number of documents found):

Abstracts of Russian Inventions (RI): 3

Applications for Russian Inventions (ZIZ): 3

Full Texts of Russian Inventions from the Last Three Bulletins (NIZ): 21

Formulas of Russian Utility Models (FPM): 0

Formulas of Russian Utility Models from the Last Three Bulletins (NPM): 0

Prospective Russian Inventions (PI): 0

Patent Search Queries for Google Patents:

1. (Intrusion Prevention System) and (machine learning) and (Intrusion detection system)
2. (Intrusion Prevention System) and (Intrusion detection system) and after:priority:20170101

Пошаговый процесс анализа патентной документации представлен в таблице 2.

Table 2 - Patent Documentation Analysis Process.

Step	Google Patents	Rospatent
Step 0 - Query	Found: 32,000	Found: 323
Step 1 - Full-text Availability Check	Remaining: 32,000	Remaining: 300
Step 2 - Title and Metadata Analysis	Remaining: 46	Remaining: 11
Step 3 - Abstract Analysis	Remaining: 21	Remaining: 7
Step 4 - Documentation Analysis and Data Extraction	Remaining: 2	Remaining: 5

1.3 Task Formulation for Management

The object of management is intrusion detection systems at any enterprise, including its components, configuration parameters, as well as the processes of detection and response to threats.

The subject of management is algorithms and methods of intelligent support for the threat detection process and ensuring the effective operation of the intrusion detection system.

Within the framework of management task formulation, the aspect of efficiency parameters of the information system is considered. The following efficiency parameters have been identified:

1. Detection Accuracy:

- The proportion of truly detected attacks and anomalies among all detected events.
- Measured by the ratio of the number of correctly classified events to the total number of detected events.

2. False Positives:

- The proportion of events incorrectly classified as attacks or anomalies among all detected events.
- Measured by the ratio of the number of false positives to the total number of detected events.

3.False Negatives:

- The proportion of actual attacks or anomalies that the system failed to detect among all real attacks or anomalies.
- Measured by the ratio of the number of undetected attacks to the total number of real attacks.

4. Response Time:

- The time required for the system to detect and respond to a threat after its occurrence.
- A shorter response time usually indicates a more efficient system.

Innovation refers to the transformation of the flow of information resources. Let's consider this transformation through the prism of the "black box" model:

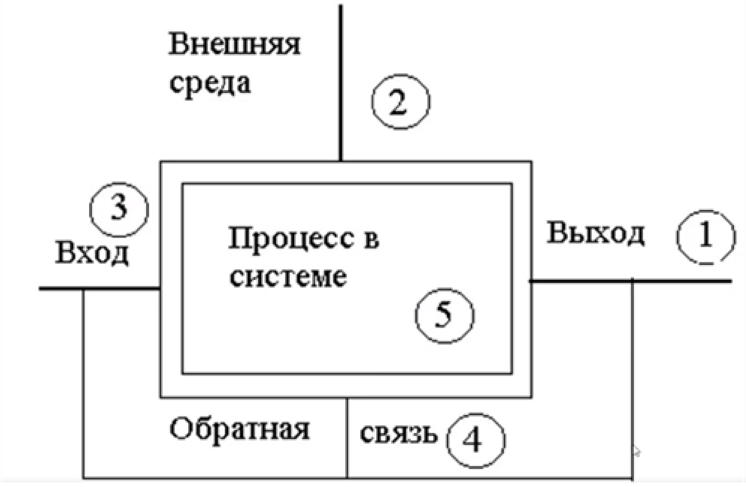


Рисунок 4 – Схематическая модель «черного ящика».

1. Input: Network traffic (data packets, network activity events), metadata about network activity (e.g., source, destination, port, protocol, etc.).

2. Output: Determination of whether the network activity is normal or abnormal (possibly indicating the type of attack), decision to block or allow network activity (in the case of IPS).
3. External environment: Network infrastructure: network nodes, routers, switches, etc. Network protocols and standards. Network topology and application architecture.
4. Feedback: In the event of detecting an attack or suspicious activity, the IDS/IPS system can generate notifications or alerts for the system administrator. In the case of blocking network activity, information about the system's actions may be sent back to logs or monitoring system for subsequent analysis or response.

An information security system can be considered as consisting of four subsystems:

1. Access control subsystem;
2. Registration and accounting subsystem;
3. Cryptographic subsystem;
4. Integrity assurance subsystem.

This research focuses on the access control subsystem.

As a supersystem, the information security department can be considered responsible for the integrity of the information system and receiving reports on the results of the intrusion detection system's operation. The supersystem coordinates the subsystem's work and interacts with the external environment.

Intrusion Detection System (IDS/IPS): This is the primary system that analyzes network activity, detects anomalies or potential attacks, and makes decisions about blocking or allowing traffic.

In the framework of the management task formulation, answers to questions of the conceptual model were provided:

1. Main function: Implementation of the continuous analysis process of network traffic and events to identify and prevent potential attacks and anomalous activity in the network.

2. System structure: Internal server-side and external interface parts of the intrusion detection and prevention system.

Internal server-side part:

- Sensors (data collection): Responsible for collecting and filtering network traffic and events.
- Analyzers (data analysis): Apply intrusion detection algorithms to analyze collected data and detect anomalies.
- Reacting devices (decision-making and response): Responsible for making decisions and taking actions to prevent threats.

3. System operation direction: Improvement of the continuous detection and prevention process of potential attacks and anomalous activity in the network, thereby ensuring the necessary security and integrity of the information infrastructure.
4. Goal: Ensuring the security of the organization's network infrastructure by detecting and blocking attempts of unauthorized access, as well as anomalous activity, which may indicate an attack or threat.

Let's define some factors that influence the value of the previously highlighted performance indicators:

1. Types of attacks and threats. Different types of attacks may have different characteristics and behavior patterns, which can affect the intrusion detection system's ability to identify them.
2. Quality of training data. The quality of data used for training machine learning algorithms or creating attack signatures can significantly affect the accuracy and reliability of detection.
3. Configuration parameters. Proper configuration of system parameters, such as thresholds for anomaly detection and false positive thresholds, can significantly affect its effectiveness.
4. Technical architecture. The system's efficiency also depends on hardware and software, network architecture, and the location of sensors.

5. Network scale. The size and scale of the network can affect the performance and capabilities of the intrusion detection system.
6. Staff training. The level of training and education of personnel responsible for configuring and monitoring the system can also significantly affect its effectiveness.
7. Degree of integration. Integrating IDS/IPS with other security systems and network devices can affect its ability to respond quickly to threats and coordinate actions.

The lifecycle of managing the operation algorithms of intrusion detection and prevention systems may include the following stages:

1. Analysis and Planning:

- Defining the requirements for the intrusion detection and prevention system.
- Studying existing algorithms and methods for threat detection and prevention.
- Planning the strategy for updating and changing algorithms according to security needs and requirements.

2. Algorithm Selection:

- Choosing the most suitable algorithms and methods for implementing the intrusion detection and prevention system.
- Taking into account security requirements, performance, and other factors.

3. Development and Implementation:

- Developing and implementing the selected algorithms and methods within the intrusion detection and prevention system.
- Integrating algorithms into the overall system architecture.
- Testing and debugging new algorithms.

4. Operation and Monitoring:

- Deploying the system in the operational environment.

- Monitoring the operation of the system and intrusion detection and prevention algorithms.
- Monitoring the performance and effectiveness of algorithms in real operational conditions.

5. Update and Adaptation:

- Conducting regular updates of algorithms according to changing threats and security requirements.
- Adapting algorithms to new types of attacks and changes in the network environment.

6. Analysis and Optimization:

- Regularly analyzing the effectiveness and performance of algorithms.
- Optimizing algorithm operation based on the results obtained and feedback from monitoring.

7. Removal and Replacement:

- In case of obsolescence or insufficient effectiveness of algorithms, removing them and replacing them with more modern or efficient alternatives.

The current state of the system can be represented as follows:

$$S_0 = \{P_0^i\}, \qquad (1),$$

where *P₀* – represents the initial values of parameters, *i = 1, 2…n* – denotes the number of parameters.

To solve the control problem, the system must reach a final state described as:

$$S_k = \{P_k^i\} \qquad (2),$$

where *Pк* – represents the final values of the system parameters.

The following parameters will be used to solve the problem:

1. Precision - measures how much the classifier can be trusted:

$$Precision = \frac{TP}{(TP+FP)} \qquad (3),$$

2. Recall - indicates how many items of the class "attack present" are correctly identified by the classifier:

$$Recall = \frac{TP}{(TP+FN)} \qquad (4),$$

3. F1-score - the harmonic mean between precision and recall (the closer to 1, the better):

$$F = \frac{(2*Precision*Recall)}{(Precision+Recall)} \qquad (5),$$

4. Accuracy - the proportion of correct answers by the algorithm:

$$Accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)} \qquad (6),$$

Notation used in the metrics formulas:

1) *TP (true positive)* - the number of true positive results,
2) *TN (true negative)* - the number of true negative results,
3) *FP (false positive)* - the number of false alarms,
4) *FN (false negative)* - the number of missed attacks.

Goal of management (*Zu*):

$$Z_u = \max K \qquad (7),$$

Where *maxK* - represents the maximization of cases of detecting unauthorized access attempts, as well as anomalous activity that may indicate an attack or threat and ensuring the security of the network infrastructure.

Control vector (u_{func}):

$$u(t) = [u_{func}] \qquad (8),$$

where u_{func} - denotes the control of operation (alteration of system operation algorithms).

In the generalized model of the control system, the controlling subsystem is the intrusion detection system (IDS/IPS).

The controlling subsystem establishes the parameters and rules of operation of the system, as well as monitors and controls its functioning.

The controlled subsystem, in turn, is the server part from which network traffic data is received into the intrusion detection system.

The scheme of the management task is presented in Figure 5.

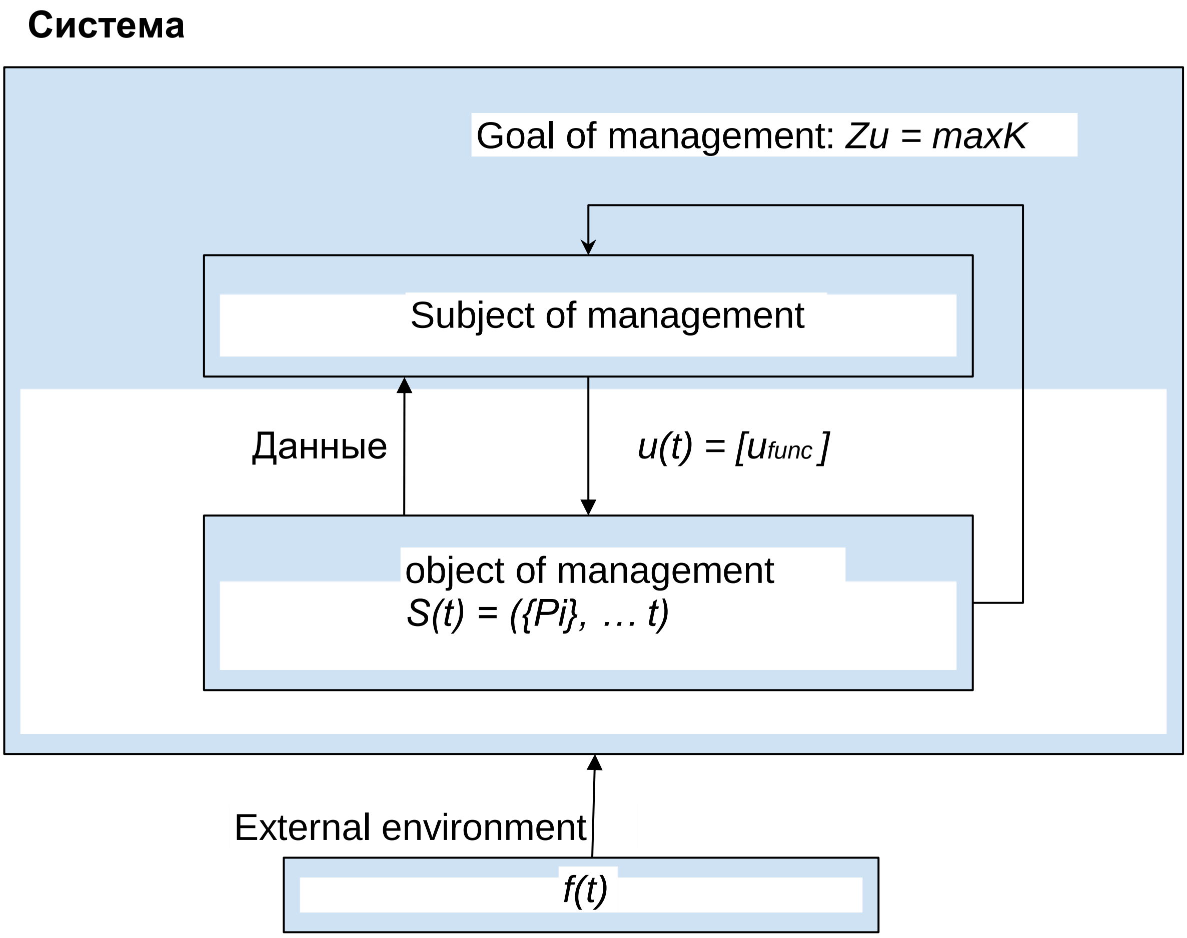


Figure 5 - Scheme of the system's management task formulation.

Results and conclusions of the first chapter

Networks play a crucial role in modern society, and cybersecurity has become a critically important area of research. Intrusion Detection Systems (IDS) monitor the operation of software in the network, but existing IDS still face challenges in increasing detection accuracy, reducing false alarms, and recognizing new types of attacks. To address these issues, many researchers focus on creating IDS that utilize machine learning algorithms.

Supervised machine learning uses labeled data to train a model, which can then predict labels for new data. For example, a spam classifier can determine whether a new email is spam or not. Unsupervised machine learning, on the other hand, works with unlabeled data. For instance, in clustering botnets attacking the network, they need to be distinguished from each other without predefined labels. Classification and regression analysis are examples of supervised machine learning, while clustering represents a form of unsupervised machine learning.

Machine learning methods can accurately identify differences between normal and anomalous data and detect unknown attacks due to their generalization ability. Cybersecurity methods mainly include antivirus software, firewalls, and Intrusion Detection Systems (IDS). Among them, IDS is a type of detection system that plays a key role in cybersecurity by monitoring the status of software and hardware operating in the network.

It is worth noting that the lack of available datasets can be the biggest challenge. Thus, unsupervised learning and incremental learning approaches have broad prospects for development. For practical IDS, interpretability is essential because interpretable models are more persuasive. The interpretability of models could be an important research direction for IDS in the future.

In the article "Applying convolutional neural network for network intrusion detection (Conference Paper)" [8], the authors concluded that Convolutional Neural Network (CNN) architectures in deep learning have achieved significant results in computer vision. To transform this performance into intrusion detection (ID) in cybersecurity, in this document, network traffic is modeled as a time series, specifically, Transmission Control Protocol/Internet Protocol (TCP/IP) packets in a predefined time range with supervised learning methods, such as Multilayer Perceptron (MLP), CNN, CNN-Recurrent Neural Network (CNN-RNN), CNN-Long Short-Term Memory (CNN-LSTM), and CNN-Gated Recurrent Unit (GRU), using millions of known good and bad connection networks.

The methodology involves measuring the effectiveness of the proposed approaches. The authors evaluated the most significant synthetic ID dataset, such as KDDCup 99. For selecting the optimal network architecture, the article conducted a comprehensive analysis of various MLP, CNN, CNN-RNN.

The models in each experiment were run for up to 1000 epochs with a learning rate in the range [0.01-05]. CNN and its architecture variations significantly outperformed classical machine learning classifiers. This was mainly because CNN has the ability to extract high-level feature representations, which are abstract forms of low-level feature sets of network traffic connections.

Deep Neural Network (DNN) is widely used for complex systems, allowing abstraction of features and learning as a machine learning method. Some researchers [9] used deep learning methodology to develop efficient and flexible IDS using one-dimensional Convolutional Neural Network (1D-CNN). The machine learning model based on 1D-CNN serialized Transmission Control Protocol/Internet Protocol (TCP/IP) packets in a specified time range as an intrusion internet traffic model for IDS, where normal and abnormal network traffic is classified and labeled for supervised learning in 1D-CNN.

As a result of comparative performance research, Random Forest (RF) and Support Vector Machine (SVM) models based on machine learning were used in addition to 1D-CNN with various network parameters and architecture. In each experiment, the models were trained for up to 200 epochs with a learning rate of 0.0001 on both imbalanced and balanced data. 1D-CNN and its architecture variations outperformed classical machine learning classifiers.

In the article "Intrusion detection using neural networks and support vector machines" [10], the authors conducted experiments on two datasets - KDD Cup 1999 and DARPA 1999, and compared the results with other intrusion detection methods such as Bayesian networks and decision trees. The authors proposed a Multilayer Perceptron (MLP) and SVM architecture for intrusion detection. The results showed that models based on MLP and SVM provide high intrusion detection accuracy, outperforming other methods.

The ensemble feature selection method improves the quality of feature selection and reduces selection time. This method was proposed by foreign researchers [11], whose goal was to develop a new approach to feature selection for intrusion detection systems. Experiments were conducted on several datasets: DARPA98, KDD99, ISC2012, and ADFA13, to evaluate the effectiveness of the proposed method. The results showed that the proposed method outperforms other feature selection methods in terms of accuracy, recall, and F-score metrics, while having lower computational complexity.

However, it is worth noting that the authors do not consider the impact of the number of selected features on the performance of the intrusion detection system. Additionally, there was no comparison with other methods based on ensemble filters, which could be an interesting aspect for future research in this area.

The "one-class SVM" algorithm (support vector machine method for anomaly detection) for creating an ensemble of models improves the efficiency of attack detection [12]. Experimental evaluation of this method was conducted on two datasets: KDDCup'99 and DARPA2000. The experiments showed that the proposed method outperforms other methods (including SVM with RBF kernel and multilayer perceptron). The analysis of the experiment results also helped identify which data characteristics affect the quality of attack detection. As a result, it was found that using different feature sets and tuning algorithm parameters can significantly affect the effectiveness of attack detection.

The article describes an intrusion detection system, a variety of intrusion detection methods to combat cybersecurity threats, which can generally be divided into signature-based intrusion detection systems (SIDS) and anomaly-based intrusion detection systems (AIDS). Some authors propose using machine learning algorithms for network traffic classification, as well as visualizing the obtained results for convenient analysis [13]. Authors used Bayesian networks, decision trees, and artificial immune system cloning algorithms as machine learning methods. The most popular publicly available datasets used for IDS research were examined, and their data collection methods, evaluation results, and limitations were discussed.

The testing was conducted only using DARPA/KDD99 datasets collected in 1999 as they are publicly available, and there are no other alternative and acceptable datasets. It is worth noting that despite their widespread recognition as a standard, these datasets no longer reflect modern "zero-day" attacks. Although the ADFA dataset contains many new attacks, it is still insufficient.

The article "Hybrid anomaly detection system for intrusion detection" [14] describes a new hybrid approach to anomaly detection for intrusion detection systems. The authors propose using statistical methods such as the maximum likelihood algorithm and decision tree-based classifiers, combined with neural networks to create an effective intrusion detection system.

The authors proposed using a variety of features, including flow information, ports, protocols, sessions, packet size, etc., as well as a variety of algorithms such as decision trees, SVM, and neural networks, to detect anomalies in network traffic.

The authors evaluated the performance of their system on the DARPA 1998 dataset, and the results showed that the hybrid approach outperforms individual methods such as SVM and neural networks in anomaly detection accuracy.

Considering the current types of attacks, one of the main problems is the "Denial of Service" (DoS) and "Distributed Denial of Service" (DDoS) attacks in a cloud environment [15]. To address this issue, using an Intrusion Detection System (IDS) as a security procedure operating at the network level is proposed. Conventional IDS in the cloud platform leads to low detection accuracy with high computational complexity. M. Mayuranathan, M. Murugan, V. Dhanakoti presented an efficient classification model based on feature subset selection for identifying DDoS attacks [15]. For DDoS attack detection in IDS, feature sets with maximum detection using the Random Harmony Search (RHS) optimization model were selected. After selecting features for DDoS detection, a deep learning-based classifier model using Restricted Boltzmann Machines (RBM) was applied. To increase the speed of DDoS attack detection, a set of seven additional layers was included in the visible and hidden layers of RBM.

As a result, accurate results are achieved through the optimization of hyperparameters of the presented deep RBM model. The probability distribution of the visible layer in the RBM model is replaced with a Gaussian distribution. For experiments, the RHS-RBM model was tested on the KDD'99 dataset.

Experimental results showed that the RHS-RBM model provides maximum accuracy - 99.92 and an F-score of 99.93. These obtained values of the RHS-RBM model were found to be better compared to the RBM model without using the RHS algorithm.

Many foreign and domestic researchers in the field of machine learning and cybersecurity conduct reviews in this area. For example, the authors of the article "A survey on machine learning techniques in wireless sensor networks intrusion detection" [16] reviewed 68 studies published from 2007 to 2014. The article discusses various machine learning methods used for intrusion detection in wireless sensor networks, such as neural networks, decision trees, support vector machines, naive Bayesian classifiers, etc.

One of the main conclusions of the article is that machine learning methods are an effective tool for intrusion detection in wireless sensor networks and can be used in combination with other methods to improve the efficiency of intrusion detection systems. It is also noted that to achieve high intrusion detection accuracy, it is necessary to consider the specific characteristics of wireless sensor networks, such as limited resources and the possibility of attacks at the physical device level.

Computer networks are constantly threatened by malicious actors who attempt to gain unauthorized access to systems on them. Malicious actors constantly refine their attack methods, while network administrators develop new defense measures in response to these threats. This ongoing interaction leads to the emergence of new vulnerabilities and exploits, as well as the removal of ineffective attack methods. Network administrators must anticipate the detection of new threats and respond to them quickly. Identifying and blocking new exploits presents a complex challenge for administrators, especially if the attack targets a small number of services on the network or has not yet gained widespread use.

LO Penchen, BRIGGS Reeves Hopp, AHMAD Navid patented an invention related to the field of network security [17]. Its technical result consists of providing more reliable and fast identification of new forms of attacks, increasing network security, and reducing processing resources used to protect the network from malicious entities. The result is achieved through a method of providing security for an online service provided over the network, using a model with continuous learning, which includes collecting a set of security signals, with the set of security signals collected in a sliding time window; identifying whether each security signal from the set of security signals is malicious or harmless; creating a balanced training dataset for the sliding time window by: ensuring a balance of malicious signals from the set of security signals based on the type of attack identified for each malicious signal, ensuring a balance of harmless signals from the set of security signals to create a balanced training dataset based on the type of device from which each harmless signal is received, and ensuring a balance of malicious signals with harmless signals by cross-connecting malicious signals with harmless signals; and creating a predictive model based on the balanced training dataset, wherein, in response to receiving an additional security signal related to a new network session from the online service, the predictive model is applied to determine whether this additional security signal is malicious or harmless [17].

By integrating continuous learning intrusion detection models into the network, the capabilities of devices and software are improved. This allows for faster and more reliable identification of new types of attacks, solving the problem of increasing network security. It also allows for more efficient use of computational resources, without spending them on detecting outdated attack methods, thereby reducing the load on the defense system against malicious actors.

To determine whether users are harmless or malicious, or whether devices are harmless (not sending malicious signals) or compromised (sending malicious signals), various security signals from the online service are collected and fed into production models to generate detection results indicating whether a given session is malicious or harmless. Security signals such as event logs, network state traces, and system commands undergo analysis using production models that evaluate their characteristics. Feature values are determined by training production models to detect malicious or safe behavior. The training data sampling mechanism excludes safe signals received from compromised devices in real-time, leaving only malicious signals from compromised devices.

The model adjusts rules or algorithms over several cycles, changing the values of variables that affect the input data to more accurately match the desired outcome. However, due to the variability of the training dataset and its large volume, achieving perfect metrics such as accuracy and precision may be unattainable.

Thus, each security signal is analyzed in a production model, which is created by training the model on a balanced dataset and tuned to determine whether a specific security signal is malicious or safe. With the active development of computer technology and networks, the task of detecting computer attacks and timely detecting cases of server infection with malicious software becomes increasingly relevant.

Network-level intrusion detection systems (IDS) use decision rule bases. These rules contain criteria for analyzing communication sessions and recording information security events. The criteria describe the content and attributes of network connections that the system considers malicious within the established syntax.

In practical application, network IDS tasks include: timely updating the decision rule base for more effective detection of new threats; reducing the number of type I errors (false positives) [21]. Kislicin N.I. considers in his patent documentation a method of autogenerating decision rules for intrusion detection systems with feedback, performed on a server, which includes at least the following steps: receiving at least one event from the event database formed by data received from at least one sensor; analyzing the received at least one event for belonging to the class of interaction with command and control centers of malware; extracting from at least one of the above events belonging to the class of interaction with command and control centers of malware, at least one feature used to form decision rules; forming decision rules using at least one of the above extracted features; saving the generated decision rules and providing the possibility of receiving updates to the decision rules for at least one sensor; sensors cyclically check the availability of updates on the central node and, if updates are available, receive them for use, in case of receiving updates on the sensors, a trigger is triggered, rebooting the decision rules [21].

The rule generation module, capable of analyzing events received from sensors, receives at least one event from the event database of the central node, received from at least one sensor, and analyzes it for belonging to the class of interaction with command and control centers of malware based on the list of identifiers of the rules of the module database. If the event belongs to such a class, the module extracts at least one feature from the event used to form decision rules.

Various identifiers can be used as features, such as the IP address of the data recipient or the domain name, which can be extracted from the service headers of transmitted data. For example, for the HTTP protocol, the domain name can be found in the Host field, and for DNS, from the binary data structure according to RFC 1035. In the case of the TDS protocol, the domain name can be extracted from the "client hello" message with the SNI extension.

After extracting features, the rule creation module checks if these characteristics are present in the list of allowed names. If they are present in this list, processing of these features is completed.

In their invention "Deep-learning-based intrusion detection method, system and computer program for web applications" [22], the authors patented an invention related to deep learning-based intrusion detection, namely, a method for detecting whether traffic is a hacker attack based on a deep neural network (DNN) model after setting up network traffic entering the server as input data to the model.

The authors suggest conducting analysis based on signatures as one of the intrusion detection methods. It represents a scheme for searching for a specific pattern corresponding to a known attack threat, and regular expressions are used to analyze strings by comparing strings with an already saved list of signatures. When a pattern with a specific signature is detected in the useful data packet, the strings are considered an attack. An accurate and limited list of signatures can reduce the number of false positives. Signature-based analysis methodology may be successful if up-to-date signature patterns are supported, however, an unknown attack, such as a zero-day attack (a security attack exploiting a security vulnerability before the vulnerability is widely known after its discovery) or the latest malware, may not be detected.

To address the above problems, the present invention provides an effective intrusion detection system through the use of deep neural networks in the form of complex web service protocol messages (Hypertext Transfer Protocol (HTTP)), which is the most common and representative for the company among various application-level services. In particular, the present invention provides a method for detecting web application threats, a system, and a computer program implementing it, configured to identify security threats by bypassing and interfering with the signature-based security detection scheme.

To achieve the above-mentioned objectives, the deep learning-based intrusion detection method for web applications according to the present invention includes: (a) inputting input data formed by preprocessing traffic data on the web server into a deep neural network model for intrusion detection; (b) outputting from the intrusion detection model information on whether intrusion is detected in the traffic data; and (c) generating an alarm signal when intrusion is detected.

In their research work titled "Cybersecurity detection and mitigation system using machine learning and advanced data correlation" [23], the authors described methods related to active security risk reduction, which are detected through combined analysis of risky users and compromised systems, a capability currently not available on the market.

Reducing attacks and risks, based on a unified view of system security as well as on identification and access constraints, can be achieved using a comprehensive Enterprise Cybersecurity Defense System (eCDS). Such an eCDS can provide the methods and design elements necessary to create a full-fledged system capable of providing active protection and mitigating unforeseen and dynamically detected cyberattacks. Such a system can be beneficial to organizations like PayPal™ as well as other individuals and corporations.

Thus, in various implementation scenarios outlined in the study, data from multiple domains (such as user identity, system logs) can be integrated into a machine learning-based solution that can recognize anomalous attempts to access electronic resources. These anomalous attempts may not be recognized by a simple rule-based system, as they could potentially be problematic. For example, a firewall might be configured to allow access to certain communication ports, or a specific user might have access to a wide range of files and relational databases. However, statistically improbable (e.g., anomalous) access attempts may still indicate a fundamental security issue, even if such access could be permitted within a rule-based system.

As a result of the research, the choice of dataset for training can be highlighted. For training attack detection systems among the available public datasets, the "Intrusion Detection Evaluation Dataset" CICIDS2017 was selected. The CICIDS2017 dataset was prepared based on the analysis of network traffic in an isolated environment, where the actions of 25 legitimate users as well as malicious actions of intruders were simulated. The dataset combines over 50 GB of "raw" data in PCAP format and includes 8 pre-processed files in CSV format containing annotated sessions with selected features observed on different days. A brief description of the files is presented in Table 3, and the quantitative composition of the dataset is provided in Table 4.

Table 3 - Brief description of files from the dataset.

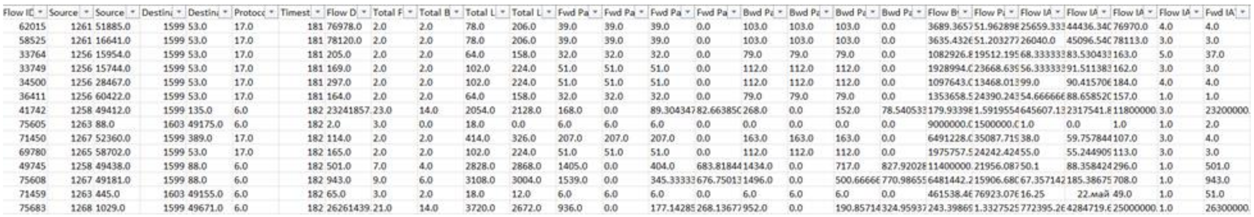
Brief description of files from the dataset		
№	File Name	Contained Attacks
1	Monday-WorkingHours.pcap_ISCX.csv	Benign (обычный трафик)
2	Tuesday-WorkingHours.pcap_ISCX.csv	Benign, FTP-Patator, SSH-Patator
3	Wednesday-workingHours.pcap_ISCX.csv	Benign, DoS GoldenEye, DoS Hulk, DoS Slowhttptest, DoS slowloris, Heartbleed
4	Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv	Benign, Web Attack – Brute Forse, Web Attack – Sql Injection, Web Attack - XXS
5	Thursday-WorkingHours-Afternoon-Infilterations.pcap_ISCX.csv	Benign, Infiltration

6	Friday-WorkingHours-Morning.pcap_ISCX.csv	Benign, Bot
7	Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv	Benign, PortScan
8	Friday-WorkingHours-Afternoon-DDoscap_ISCX.csv	Benign, DDoS

Table 4 - Quantitative composition of the dataset.

Quantitative composition of the dataset		
№	Record Type	Number of Records
1	BENING	2359087
2	DoS Hulk	231072
3	PortScan	158930
4	DDoS	41835
5	DoS GoldenEye	10293
6	FTP-Patator	7938
7	SSH-Patator	5897
8	DoS slowloris	5796
9	DoS Slowhttptest	5499
10	Bot	1966
11	Infiltration	36
12	Heartbleed	11
13	Web Attack – Brute Force	1507
14	Web Attack – XSS	652
15	Web Attack – SQL Injection	21

At Figure 6, a fragment from the dataset is presented.



Flow ID	Source	Source	Destin	Destin	Protocc	Timest	Flow D	Total F	Total B	Total L	Total L	Fwd Pa	Fwd Pa	Fwd Pa	Fwd Pa	Bwd Pa	Bwd Pa	Bwd Pa	Bwd Pa	Flow B	Flow P	Flow IA	Flow IA	Flow IA	Flow IA	Fwd IA
62015	1261	51885.0	1599	53.0	17.0	181	76978.0	2.0	2.0	78.0	206.0	39.0	39.0	39.0	0.0	103.0	103.0	103.0	0.0	3689.3657	51.96289E	25659.333	44436.34C	76970.0	4.0	4.0
58525	1261	16641.0	1599	53.0	17.0	181	78120.0	2.0	2.0	78.0	206.0	39.0	39.0	39.0	0.0	103.0	103.0	103.0	0.0	3635.432€	51.203277	26040.0	45096.54C	78113.0	3.0	3.0
33764	1256	15954.0	1599	53.0	17.0	181	205.0	2.0	2.0	64.0	158.0	32.0	32.0	32.0	0.0	79.0	79.0	79.0	0.0	1082926.8	19512.195	68.333333	83.530433	163.0	5.0	37.0
33749	1256	15744.0	1599	53.0	17.0	181	169.0	2.0	2.0	102.0	224.0	51.0	51.0	51.0	0.0	112.0	112.0	112.0	0.0	1928994.C	23668.639	56.333333	91.511383	162.0	3.0	3.0
34500	1256	28467.0	1599	53.0	17.0	181	297.0	2.0	2.0	102.0	224.0	51.0	51.0	51.0	0.0	112.0	112.0	112.0	0.0	1097643.C	13468.013	99.0	90.415706	184.0	4.0	4.0
36411	1256	60422.0	1599	53.0	17.0	181	164.0	2.0	2.0	64.0	158.0	32.0	32.0	32.0	0.0	79.0	79.0	79.0	0.0	1353658.5	24390.243	54.666666	88.65852C	157.0	1.0	1.0
41742	1258	49412.0	1599	135.0	6.0	182	23241857.	23.0	14.0	2054.0	2128.0	168.0	0.0	89.304347	82.66385C	268.0	0.0	152.0	78.540533	179.93398	1.5919554	645607.13	2317541.8	11800000.	3.0	23200000.
75605	1263	88.0	1603	49175.0	6.0	182	2.0	3.0	0.0	18.0	0.0	6.0	6.0	6.0	0.0	0.0	0.0	0.0	0.0	9000000.C	1500000.C	1.0	0.0	1.0	1.0	2.0
71450	1267	52360.0	1599	389.0	17.0	182	114.0	2.0	2.0	414.0	326.0	207.0	207.0	207.0	0.0	163.0	163.0	163.0	0.0	6491228.C	35087.719	38.0	59.757844	107.0	3.0	4.0
69780	1265	58702.0	1599	53.0	17.0	182	165.0	2.0	2.0	102.0	224.0	51.0	51.0	51.0	0.0	112.0	112.0	112.0	0.0	1975757.5	24242.424	55.0	55.244909	113.0	3.0	3.0
49745	1258	49438.0	1599	88.0	6.0	182	501.0	7.0	4.0	2828.0	2868.0	1405.0	0.0	404.0	683.81844	1434.0	0.0	717.0	827.92028	11400000.	21956.087	50.1	88.358424	296.0	1.0	501.0
75608	1267	49181.0	1599	88.0	6.0	182	943.0	9.0	6.0	3108.0	3004.0	1539.0	0.0	345.33333	676.75013	1496.0	0.0	500.66666	770.98655	6481442.2	15906.68C	67.357142	185.38675	708.0	1.0	943.0
71459	1263	445.0	1603	49155.0	6.0	182	65.0	3.0	2.0	18.0	12.0	6.0	6.0	6.0	0.0	6.0	6.0	6.0	0.0	461538.4€	76923.07€	16.25	22.май	49.0	1.0	51.0
75683	1268	1029.0	1599	49671.0	6.0	182	26261439.	21.0	14.0	3720.0	2672.0	936.0	0.0	177.14285	268.13677	952.0	0.0	190.85714	324.95937	243.39865	1.3327525	772395.26	4284719.6	25000000.	1.0	26300000.

Figure 6 - Dataset for the research.

For further analysis, the following 10 most common machine learning models (algorithms) were selected for comparison (abbreviations are provided in parentheses along with the corresponding implementation of the model from the scikit-learn package):

1. K-Nearest Neighbors (KNN, sklearn.neighbors.KNeighborsClassifier).
2. Support Vector Machine (SVM, sklearn.svm.SVC).
3. Decision Tree (CART, CART learning algorithm, sklearn.tree.DecisionTreeClassifier).
4. Random Forest (RF, sklearn.ensemble.RandomForestClassifier).
5. AdaBoost (AdaBoost, sklearn.ensemble.AdaBoostClassifier).
6. Logistic Regression (LR, sklearn.linear_model.LogisticRegression).
7. Naive Bayes (NB, sklearn.naive_bayes.GaussianNB).
8. Linear Discriminant Analysis (LDA, sklearn.discriminant_analysis.LinearDiscriminantAnalysis).
9. Quadratic Discriminant Analysis (QDA, sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis).
10. Multi-layer Perceptron (MLP, sklearn.neural_network.MLPClassifier).

From the literature review on vulnerability research of intrusion detection systems (IDS/IPS) based on statistical methods and machine learning methods, the following conclusions can be drawn:

1. Analyzing intrusion detection algorithms such as XGBoost, KNN, SVM, and others reveals the diversity of approaches to cybersecurity problem-solving. Assessing the effectiveness of these algorithms across various types of attacks demonstrates the potential for enhancing the detection and prevention of new network threats.

2. Research into scientific papers and patents indicates the ongoing development of intrusion detection methods based on statistical techniques and machine learning, emphasizing the relevance and significance of this topic for cybersecurity.
3. The objective of evaluating IDS/IPS based on statistical methods and machine learning is to enhance the level of protection for information systems against cyberattacks. Investigating vulnerabilities in IDS/IPS systems and identifying common attack types helps to better understand weaknesses in existing systems and develop more reliable defense mechanisms.
4. The project tasks, such as algorithm analysis, test environment development, algorithm effectiveness assessment, and approach comparison, provide a deep understanding of cybersecurity issues and identify the most effective protection methods. The research findings can be utilized by cybersecurity professionals to enhance intrusion detection systems in both industrial and academic settings.

Thus, the analysis of intrusion detection algorithms based on statistical methods and machine learning indicates a low level of research depth in the subject matter and the potential for significant improvement in cybersecurity through the application of modern methods and technologies.

2 COMPARATIVE ANALYSIS OF EXISTING MACHINE LEARNING METHODS FOR ASSESSING IDS/IPS VULNERABILITIES

2.1 Network Anomalies and Methods for Their Detection

2.1.1 Types of Network Anomalies

Currently, one of the actively developing and demanded directions in the field of information security is the detection of attacks and prevention of intrusions by malicious actors into computer systems and corporate networks. To achieve this, a range of specialized algorithms and tools are applied, using behavior models and signature methods to detect known and unknown attacks and identify anomalous activities. This approach is highly effective in detecting insider attacks and "zero-day" attacks.

An anomaly is a deviation or divergence from a rule, so anything deviating or diverging from what is correct or normal is considered anomalous.

When detecting a network anomaly, in order to make decisions about further actions, it is necessary to carefully study its nature, potential danger, and possible consequences, i.e., to solve a classification problem. In this work, a generalized approach to the classification of network anomalies is proposed (Figure 7).

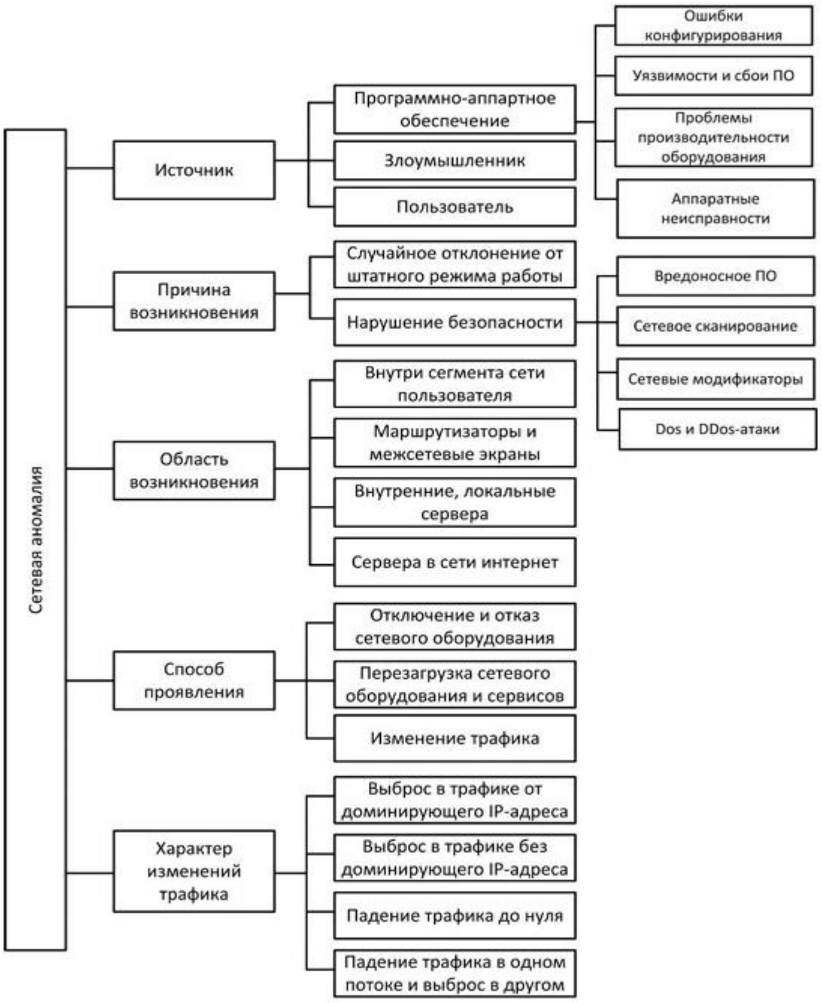

Сетевая аномалия
Источник
Программно-аппартное обеспечение
Злоумышленник
Пользователь
Ошибки конфигурирования
Уязвимости и сбои ПО
Проблемы производительности оборудования
Аппаратные неисправности
Причина возникновения
Случайное отклонение от штатного режима работы
Нарушение безопасности
Вредоносное ПО
Сетевое сканирование
Сетевые модификаторы
Dos и DDos-атаки
Область возникновения
Внутри сегмента сети пользователя
Маршрутизаторы и межсетевые экраны
Внутренние, локальные сервера
Сервера в сети интернет
Способ проявления
Отключение и отказ сетевого оборудования
Перезагрузка сетевого оборудования и сервисов
Изменение трафика
Характер изменений трафика
Выброс в трафике от доминирующего IP-адреса
Выброс в трафике без доминирующего IP-адреса
Падение трафика до нуля
Падение трафика в одном потоке и выброс в другом

Figure 7 – generalized approach to the classification of network anomalies

As the main classification features, the following are used:

1. Source type;
2. Cause of occurrence;
3. Area (location) of occurrence;
4. Manifestation method;
5. Nature of changes.

In this regard, for identifying potential network attacks, the most significant features would include the source of occurrence, the area of manifestation, and the nature of traffic changes. Table 4 presents a description of the relationship between anomalies classified by the cause of occurrence and the nature of network traffic changes.

Table 4 - Description of network traffic anomalies.

Type and cause of network anomaly	Description	Traffic change characteristics
Alpha Anomaly	Extremely high point-to-point traffic level	Spike in traffic representation in bytes/s, packets/s for a dominating source-destination flow. Duration around 10 minutes.
DoS-, DDoS-	Distributed denial-of-service attack on a single victim	Spike in traffic representation in packets/s, flows/s, from multiple sources to a single destination address.
Overload	Unusually high demand on a single network resource or service	Spike in traffic per flows/s to a dominating IP address and port. Typically a short-term anomaly.
Network/Port Scanning	Network scanning for specific open ports or host scanning for all ports to identify vulnerabilities	Spike in traffic per flows/s, with multiple packets in streams originating from a single dominating IP address.
Worm Activity	Malicious software capable of self-propagation across networks and exploiting OS vulnerabilities	Spike in traffic without a dominating destination address, but always with one or several dominating destination ports.
Point-to-Multipoint	Content distribution from one server to multiple users	Spike in bytes from a main source to multiple destinations, to a well-known port.
Outages	Network disruptions causing a drop in traffic between a source and destination pair	Decrease in packet, flow, and byte traffic typically down to zero. May be long-term and include all traffic flows from or to a single router.
Flow switching	Unusual switching of traffic flows from one incoming router to another	Drop in bytes or packets in one traffic flow and an increase in another. May involve multiple traffic flows.

2.1.2 Methods of anomaly detection

Passive Network Monitoring: The computer network includes sensors that collect data from the network and evaluate it. In this scenario, there are two possibilities. The collected data may be intended directly for the sensors (for example, events sent via the SNMP protocol), or it may be a copy of the production traffic occurring in the network regardless of whether a sensor is connected or not.

Active Network Monitoring: The network may also contain sensors that generate additional traffic, which they send through the network. This traffic can be used to continuously determine the availability or general parameters of the tested services, network lines, and devices.

Accordingly, methods of anomaly detection in these categories can be divided into four broad groups: behavioral methods, machine learning methods, computational intelligence methods, and knowledge-based methods.

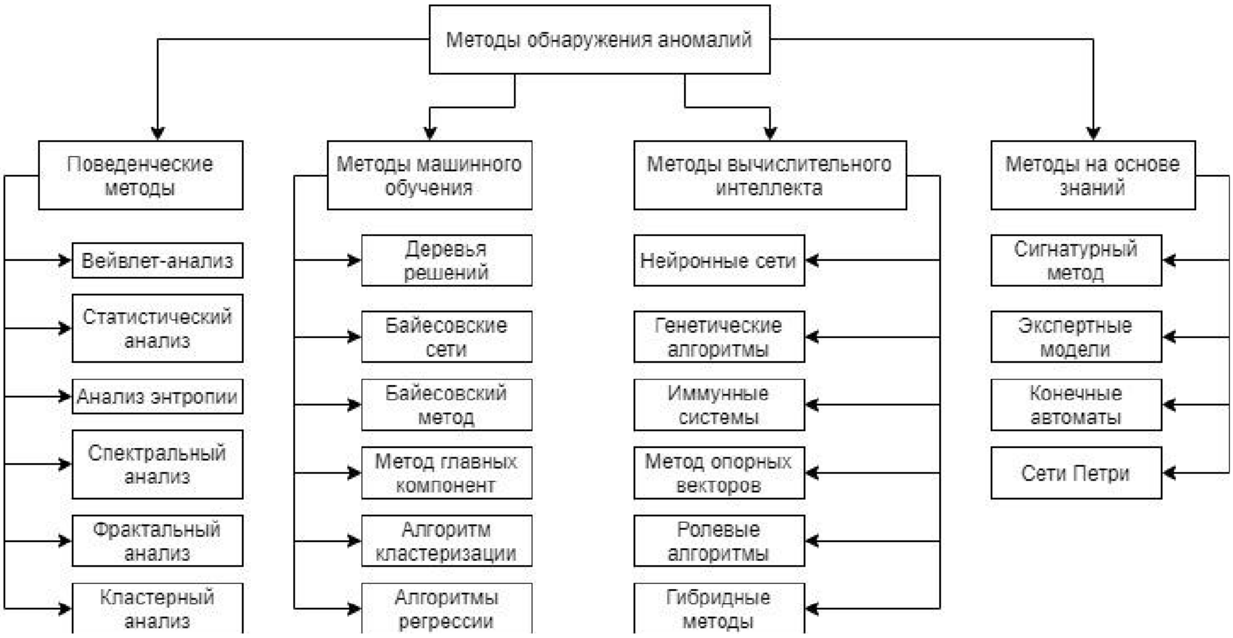


Figure 8 - Anomaly Detection Methods

2.1.2.1 Behavioral Methods

Wavelet transformation of a one-dimensional signal involves decomposing it into a basis constructed from a soliton-like function (wavelet) with certain properties, using scale changes and translations. Performing wavelet transformation allows for a clearer distinction of the signal component with greater amplitude and reduces the influence of small amplitudes, which mostly represent noise components of the signal.

Statistical analysis is a part of behavioral methods for intrusion detection and is based on comparing the current state of the network with predefined features characterizing the normal state of the network. The major challenge lies in attacks with anomalous behavior in the header of selected packet telecommunications traffic. The application of statistical analysis methods is the most common way to implement anomaly detection technology.

For real-time analysis of anomalous intrusions, the following current statistical characteristics are calculated: sample mean, sample variance, skewness coefficient, and kurtosis coefficient. The detection process occurs in two stages. In the first stage, training is performed, assuming the absence of anomalous outliers in the observed interval. During this period, the threshold for anomaly detection is determined based on a specified probability of Type I error. The key moment for accurate detection is the correct selection of the training segment, where the threshold level is established. Then, according to the proposed methodology, a decision is made about the presence or absence of an attack by analyzing data in a sequentially shifting window. At each window position, the analysis is based on a sequential data analysis.

Statistical analysis methods can be divided into two main groups:

1. Parametric methods: Assume that normal data is generated by a parametric distribution with parameters θ and probability density function P(x,θ), where x is the observation. An anomaly is the inverse function of the distribution. These methods are often based on Gaussian or regression models, as well as their combinations.

2. Non-parametric methods: It is assumed that the model structure is not predefined, but is determined from the provided data. This category includes methods based on histograms or kernel functions.

Anomaly detection systems based on the concept of "entropy" analyze network flows rather than individual network packets. Network flows represent one-way metadata about packets with the same source and destination IP addresses, ports, and IP protocol type. It is important to note that all network activity at OSI model levels 3 and above is reduced to flows, including not only TCP connections but also stateless protocols such as UDP and ICMP. The advantages of using the concept of flows include the following:

- They require minimal resources for usage and storage, facilitating analysis.
- They pose fewer problems with confidentiality and protection of personal data.
- Access to the necessary information in the network is easily organized, for example, through Cisco NetFlow, sFlow, or IPFIX.

Spectral methods find data approximation using a combination of attributes that capture most of the variability in the data. This methodology is based on the assumption that data can be embedded in a lower-dimensional subspace where normal states and anomalies manifest differently. Spectral methods are often used in conjunction with other algorithms for data preprocessing. Modifications of spectral methods are investigated in the work by V.P. Shkodyrev, K.I. Yagafarov, V.A. Bashtovenko [24].

Fractal analysis methods allow timely detection of anomalous traffic. The main parameter of fractal analysis is the Hurst exponent (scaling exponent). It is most commonly used in time series analysis. The larger the delay between two identical pairs of values in a time series, the smaller the Hurst exponent. The hypothesis is advanced that to find the Hurst exponent, it is sufficient to know whether the process under study is stationary or not. The choice of algorithm for further computation of this exponent depends on this. It should be noted that there are few practical experiments aimed at studying the fractal properties of traffic.

All statistical analysis methods have similar drawbacks. Firstly, malicious software adapts to the behavior of ordinary users, which reduces the effectiveness of statistical methods. Secondly, it is difficult to establish a threshold that allows for effective detection of anomalies and intrusions with minimal false positives. In addition, statistical methods require complete information about the processes taking place, which is challenging in conditions of limited data.

2.1.2.2 Machine Learning Methods

The reason for using machine learning is that it can help automate threat processing and continuously update the system by analyzing threats and recognizing them. In other words, the software learns to recognize traffic patterns in order to classify different events and either reject or allow traffic.

Machine learning is the ability of a program or system to learn and improve its functions based on the tasks assigned. Unlike statistical methods, which focus on understanding the process itself, machine learning involves creating a system that evolves based on accumulated knowledge. Machine learning-based systems can adjust their data processing strategy in response to new information. However, machine learning methods require significant computational resources, and adapting them to specific domains can be challenging. The ML approach typically consists of the following stages:

- Defining class attributes (features) and the classes themselves in the training data.
- Determining a subset of attributes needed for classification (i.e., dimensionality reduction).
- Training the model using training data.
- Using the trained model to classify unknown data in testing mode.

Depending on the type of data classes used to implement the algorithm, anomaly detection methods can be performed in one of the following three modes:

1. Supervised anomaly detection: This method requires a training set that fully represents the system and includes instances of both normal and anomalous data classes. The algorithm operates in two stages: training and recognition. During training, a model is built, which is then used to compare unlabeled instances. In most cases, it is assumed that the data does not change its statistical characteristics; otherwise, there is a need to modify the classifier.
2. Semi-Supervised anomaly detection: In this approach, the original data represents only the normal class. After being trained on one class, the system

can determine whether new data belongs to it, thereby identifying anomalies. Algorithms operating in semi-supervised mode do not require information about the anomalous class of instances, making them more widely applicable and capable of detecting deviations without predefined information about them.

3. Unsupervised anomaly detection: This method is applied when there is no prior information about the data. Unsupervised anomaly detection algorithms assume that anomalous instances occur much less frequently than normal ones. Data is processed, and the most distant points are identified as anomalies. This methodology requires access to the entire dataset and cannot be applied in real-time mode.

Decision trees are a non-parametric supervised learning method used for classification and regression tasks. The main goal is to create a model that predicts the value of the target variable using simple decision rules derived from the features of the data. Decision trees can be considered as piecewise-constant approximations.

A Bayesian network is a graphical model representing probabilistic dependencies between a set of variables, allowing for probabilistic inference using these variables. It consists of two main components: a graphical structure that defines dependencies and independencies between random variables representing the domain, and a set of probability distributions defining the strength of dependencies encoded in the graphical structure. In the context of anomaly detection, Bayesian networks are used to estimate the probability of an observation belonging to one of the normal or anomalous classes. The simplest implementation of this approach is the naive Bayes approach.

A clustering algorithm involves grouping similar instances into clusters and does not require knowledge of the properties of potential anomalies. Anomalies detection can be based on the following assumptions: - Normal data instances belong to a data cluster, while anomalies do not belong to any of the clusters. However, this formulation may encounter a problem of defining precise cluster boundaries. Hence, another assumption follows: - Normal data are closer to the center of the cluster, while anomalies are significantly farther away. In cases where anomalous instances are not singular, they can also form clusters. Thus, their detection is based on the following assumptions:

- Normal data form large dense clusters, while anomalies form small and sparse ones.
- Normal objects are close to the cluster center, while anomalies are distant from the center.
- Normal objects belong to large, dense clusters, while anomalies belong to small and sparse ones.

One of the simplest implementations of the clustering-based approach is the k-means algorithm.

2.1.2.3 Methods of artificial intelligence

Methods of computational intelligence include the use of artificial neural networks, immune networks, genetic and swarm algorithms, support vector machines, and other approaches for intrusion detection. Except for the support vector machine method, all these methods are based on "peeking" at the chains of actions of living beings (individual organisms or populations) and translating them into a mathematical language. In other words, machines implement algorithms that simulate phenomena of living nature, which react more flexibly to the environment and execute faster computationally.

Neural networks can make inferences about new objects based on incomplete data, classifying them into appropriate attack categories. Similar to living beings, they can both make mistakes and correctly guess, depending on the quality of training and the training dataset. The ability to self-learn eliminates the need for constant signature updates, reduces the system's response time to network anomalies, and allows processing a larger volume of traffic, thereby increasing the level of information security.

More complex, though similar, method is artificial immune networks (AINs), based on the human immune system. Typically, algorithms such as negative selection and clonal selection are used for their training. The immune system is a distributed multilevel defense mechanism against foreign microorganisms, viruses, and pathogens. Each level of immunity performs its type of defense reaction, and the higher the level, the higher the specificity of the response.

The negative selection algorithm is based on the mechanism of T-lymphocyte maturation in the thymus. Input data for this algorithm is a set of strings composed of characters from a specific alphabet (e.g., numbers or letters).

The clonal selection algorithm, belonging to the class of evolutionary algorithms, is used to solve optimization problems. The key concept of this algorithm is affinity, which in immunology means the degree of compatibility between two cells, and in mathematical implementation, it represents the value of the optimized function. During the algorithm's operation, a population of antibodies P is generated, representing a set of randomly created arguments of the optimized function. Then, the affinity of each antibody is calculated. After that, each antibody is cloned, and the higher its affinity, the more clones are created. Then, each antibody (including clones) undergoes mutation, with the lower the affinity of the antibody, the more mutations occur. Mutation involves introducing random changes into the elements of the antibody. After mutation, the affinity of each antibody is recalculated, and as a result, n antibodies with the best affinity are selected. These antibodies are added to the memory cell pool M. Then, n worst antibodies from the initial population P are replaced by antibodies from M. In the proposed approach, the clonal selection algorithm is used to improve the quality of attack detection and reduce the level of false positives.

The support vector machine (SVM) method is applied for anomaly detection in systems where normal behavior is represented by only one class. This method defines the boundary of the region where the instances of normal data are located. For each examined instance, it is determined whether it is in a certain region. If the instance is outside the region, it is identified as anomalous.

2.1.2.4 Knowledge-based methods

Knowledge-based methods include approaches that utilize predefined facts, inference rules, and pattern matching to detect anomalies (attacks) based on an embedded search mechanism. Search procedures may involve pattern matching, regular expressions, state transition analysis, and other methods. These methods are so named because systems employing them operate with a knowledge base containing descriptions of known attacks.

The knowledge base serves as a repository with expert-contributed records supporting data processing logic and interpretation and includes a logical inference subsystem.

The signature-based method can protect against viral or hacker attacks if the attack signature is already known and entered into the system's attack detection database (AD). However, during the first encounter with an unknown virus, when the attack signature is absent from the database, a signature-based AD will fail to recognize the threat and consider it legitimate. Such vulnerabilities are referred to as zero-day vulnerabilities. This approach, akin to virus detection technologies, enables the system to detect all known attacks but is incapable of recognizing new, as yet unknown types of attacks.

This method is straightforward to implement and forms the basis of most intrusion detection systems. However, administrators encounter several challenges when operating such systems. The first challenge lies in creating a mechanism for describing signatures, that is, a language for describing attacks. The second problem, related to the first, is correctly describing the attack to capture all possible variations.

2.1.3 Network traffic analysis

The most likely way for attackers to penetrate infrastructure is through interception via the network environment, which is a system of connections between nodes for data transmission. Network security involves a wide range of measures to protect computer networks and endpoints from malicious activities, misuse, and critical failures.

Network firewalls are perhaps the most widely known network security tools, which use access strategies and unauthorized traffic filtering between devices in the network environment. However, network security is not limited solely to the use of network firewalls.

Due to the variety of potential threats and the multitude of possible attacks, the network security model is a complex system. Administrators need to counter attacks from different directions and not rely solely on one component to ensure security. The interaction between clients and the network begins with access control, which is an authorization method that allows administrators to control user, role, or device access to various parts of the network.

Intrusion detection systems operate within the network and are used to detect attempts or successful attacks through passive observation. Intrusion prevention systems (IPS) are an evolution of intrusion detection systems (IDS), allowing interception and analysis of traffic between the source and destination for automatic anomaly detection. Intercepting and analyzing network packets in real-time, known as sniffing, is considered a critical requirement for intrusion detection and prevention systems as it provides access to the content and data passing through the network, helping to identify threats.

Considering the possibility that attackers can bypass access control measures and evade detection by intrusion detection systems, it is important to anticipate the likelihood of network infiltration. Well-designed systems should be prepared to detect insider attacks. Administrators need to actively use network activity monitoring and logging tools to expand visibility on servers and between them. Restricting protection only at the perimeter is insufficient, as attackers who overcome this barrier often succeed. Proper network segmentation can limit damage.

Micro-segmentation is the practice of dividing the network into different sections based on the functionality of each element. When properly configured, micro-segmentation simplifies network structure and security strategy management.

However, its effectiveness depends on clearly defined infrastructure change processes. Changes in the network must be accurately reflected in the micro-segmentation schemes, which can be a challenging task. Nevertheless, network segmentation allows administrators to strictly control and manage different routes between nodes A and B, and provides an extended visibility area for applying data analysis methods to detect attacks.

Intercepting data transmitted over the network is a key method for recording network activity for subsequent online and offline analysis. Similar to a surveillance camera at a crossroads to monitor traffic, packet analyzers (sniffers) intercept and record traffic on the network. Network activity logs are useful not only for security incident investigation but also for debugging, performance monitoring, and network operations control. Positioned at strategic points in the network and properly configured packet analyzers can become an important tool for creating detailed data sets that provide a comprehensive view of what is happening on the network.

2.2 Comparative analysis of well-known machine learning algorithms applied in IDS/IPS.

The machine learning algorithm operates by processing the training dataset and creating a model. The model, in turn, uses new data for predictions, maintaining the format of the training data. All machine learning algorithms comprise three key components: a model family, which defines possible model variations; a loss function, which numerically evaluates the quality of models; and an optimization procedure, which selects the best model from the given family.

After setting constraints on the selection of forecasting algorithms in a specific parameterized family, it is necessary to choose the optimal algorithm for the training dataset. However, how can one ensure that the best algorithm is chosen? The best algorithm should optimize the numerical metric computed based on the studied data. This metric is called the objective function. In the context of machine learning, the objective function is also referred to as the cost function or loss function, as it helps quantitatively assess the "cost" of incorrect predictions or associated losses.

From a mathematical perspective, the loss function is a function that maps pairs of values (predicted label, truth label) to a number. The goal of the machine learning algorithm is to find such model parameters that minimize the loss function, which is obtained by processing the training data. An optimization algorithm is used to implement the search process.

Optimization algorithms are divided into two main groups:

1. First-order optimization algorithms - they use the first derivative of the objective function with respect to the model parameters for its optimization. Gradient descent methods are the most common type of such algorithms. They are used to find the minimum or maximum value of the objective function by computing the gradient of the function, i.e., partial derivatives with respect to each parameter. The gradient determines the direction in which parameter values should be adjusted to achieve the most optimal result provided by the function.
2. Second-order algorithms, also known as second-order methods, use the second derivatives of the objective function for optimization. Unlike first-order algorithms, they have a higher convergence rate and can successfully solve saddle point problems. However, second-order methods typically require more computational resources and may be slower compared to first-order algorithms.

The choice of a suitable optimization algorithm depends on the size of the dataset, the type of learning task, and the requirements for the necessary resources.

The first-order optimization algorithm group includes:

1. LIBLINEAR [27] - the default solver for linear classification in the scikit-learn library. This algorithm is not very efficient for large datasets, so the scikit-learn documentation recommends using Stochastic Average Gradient (SAG) or SAGA (an improved version of SAG) methods, which perform better with large datasets.
2. Stochastic Gradient Descent (SGD) - a simple and efficient optimization algorithm that updates parameters for each individual training data instance. The stochastic nature of gradient descent means that this algorithm is more likely to

find new and possibly better local minima compared to the standard gradient descent method.

3. AdaGrad, AdaDelta, and Adam (Adaptive Moment Estimation) - these algorithms allow for the separation and adaptation of learning rates for each parameter and solve some tasks with other simpler gradient descent algorithms.

As in many areas of data science, there is no universally ideal optimization algorithm. Determining the best algorithm for specific tasks often requires trial and error. Selection criteria include not only convergence and speed but also other factors. It is often customary to start with the default or most reasonable option and gradually improve it.

Ensemble learning involves combining multiple classifiers to create a more complex and often more effective classifier. Combining decision trees into ensembles is a widely used method for creating high-quality classifiers. These ensembles are often referred to as decision forests. The most common types in practice are random forests and gradient-boosted decision trees.

Random forests consist of simple ensembles of multiple decision trees, which typically contain tens to thousands of such trees. The presence of multiple decision trees in the forest leads to a high degree of similarity between trees and a large number of repeated splits in the trees, especially for features that are the most stringent predictors of the dependent variable. The algorithm for constructing a random forest solves this problem.

Gradient-boosted decision trees (GBDT) apply more complex combinations of predictions from individual decision trees. When using the gradient boosting methodology, several weak learners are selectively combined by performing gradient descent optimization in the loss function to obtain a much more powerful learning model.

GBDT has been enhanced to improve performance, enhance generalization, and create more efficient models. Some of these improvements are highlighted below:

1. Setting artificial constraints for trees (e.g., limiting tree depth, maximum number of nodes, or minimum number of elements in a node) helps limit the capabilities of trees without compromising their learnability.
2. Sometimes decision trees added at early stages of training gradient-boosted ensembles may have a greater impact on the overall prediction than those added later. This can lead to model imbalance, reducing the benefits of ensembling. To address this issue, a weighted assessment of each tree's contribution is applied to slow down the learning process. A "shrinkage" technique is also used to reduce the influence of individual trees, allowing later trees to improve the model.
3. Properties of random forests, which are based on stochastic processes, can be combined with gradient boosting methodology by applying data subsampling before building trees and by thinning the feature set before using it for branching.
4. To prevent overfitting, widely used regularization methods such as L1 and L2 regularization are applied to balance the learning weights.

XGBoost is a widely used gradient boosting method for decision trees that delivers outstanding results when working with large volumes of data while maintaining the ability to scale correctly [28]. It serves as the foundation for many innovative ideas in machine learning and has attracted attention from the community as a reliable solution for creating decision tree ensembles. However, GBDT is more prone to overfitting compared to regular random forests and is more difficult to parallelize due to its additive training, which depends on the results of each tree when updating the gradient for the next one.

Based on the original typical scheme (Figure 9), the following action plan can be developed for the development of an attack detection system, which complements the signature analyzer to increase the overall efficiency of the system, especially regarding previously unknown attacks:

1. Selecting a dataset for training the computer attack detection system.
2. Preprocessing the data.
3. Sampling against class imbalance.

4. Assessing feature importance and selection.
5. Reducing the feature space.
6. Choosing a model.
7. Tuning and training the model.
8. Testing and validation.

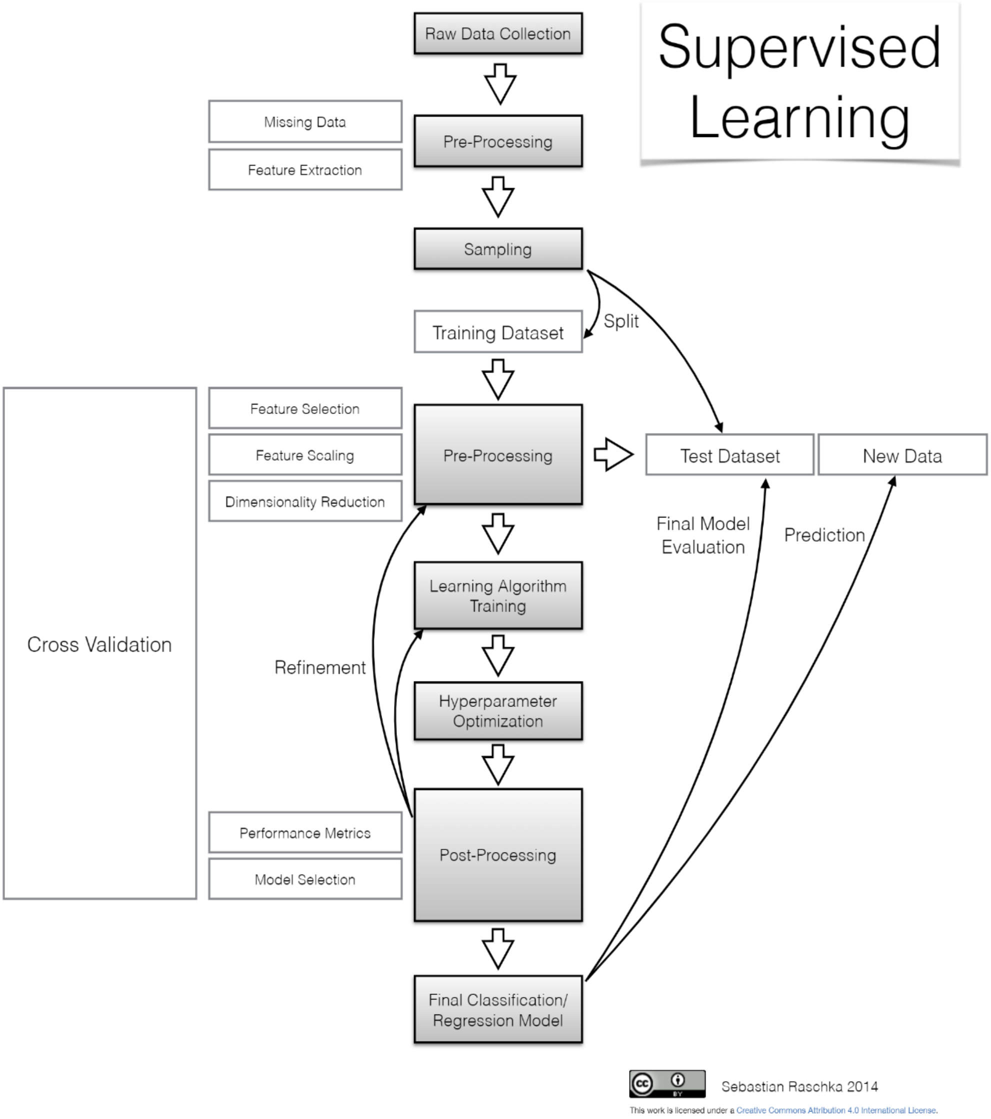


Figure 9 - Typical Supervised Learning Scheme by Sebastian Raschka (licensed under CCA 4.0)

2.2.1 Selecting a dataset for training the computer attack detection system

Some of the available datasets suffer from a lack of diversity and traffic volume, some do not cover the variety of known attacks, while others anonymize packet payload data, which may not reflect current trends. Some also lack feature sets and metadata. For training the attack detection system among the available public datasets (DARPA1998, KDD1999, ISCX2012, ADFA2013, and others), one of the current and comprehensive ones was chosen - the "Intrusion Detection Evaluation Dataset" CICIDS2017. Developed by the Canadian Institute for Cybersecurity.

The CICIDS2017 dataset is prepared based on the analysis of network traffic in an isolated environment, where the actions of 25 legitimate users and malicious actions of intruders were modeled.

The CICIDS2017 dataset contains safe and modern common attacks that resemble real-world data (PCAP). It also includes the results of network traffic analysis using CICFlowMeter with flow marking based on timestamps, source and destination IP addresses, source and destination ports, protocols, and attacks (CSV files).

Creating realistic background traffic was the main priority for the developers when creating this dataset. They used their proposed B-profile system (Sharafaldin et al., 2016) to profile the abstract behavior of human interactions and generate naturalistic background traffic. For this dataset, abstract behavior of 25 users was constructed based on HTTP, HTTPS, FTP, SSH, and email protocols.

The dataset combines over 50 GB of "raw" data in PCAP format and includes 8 preprocessed CSV files containing labeled sessions with selected features on different observation days. The comma-separated values (CSV) format in this dataset is a standard way of representing data for analytical research.

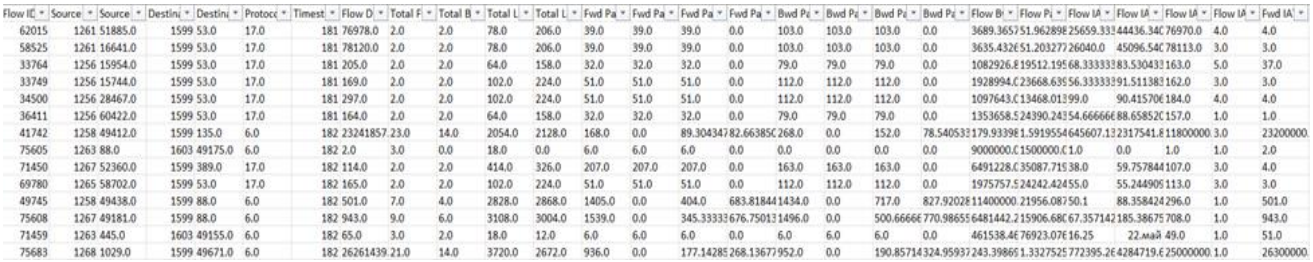
A brief description of the files and the quantitative composition of the dataset are presented in the tables and figures below.

Table 5 - Brief Description of Files in the Dataset

Brief Description of Files in the Dataset		
№	File Name	Contained Attacks
1	Monday-WorkingHours.pcap_ISCX.csv	Benign
2	Tuesday-WorkingHours.pcap_ISCX.csv	Benign, FTP-Patator, SSH-Patator
3	Wednesday-workingHours.pcap_ISCX.csv	Benign, DoS GoldenEye, DoS Hulk, DoS Slowhttptest, DoS slowloris, Heartbleed
4	Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv	Benign, Web Attack – Brute Forse, Web Attack – Sql Injection, Web Attack - XXS
5	Thursday-WorkingHours-Afternoon-Infilterations.pcap_ISCX.csv	Benign, Infiltration
6	Friday-WorkingHours-Morning.pcap_ISCX.csv	Benign, Bot
7	Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv	Benign, PortScan
8	Friday-WorkingHours-Afternoon-DDoscap_ISCX.csv	Benign, DDoS

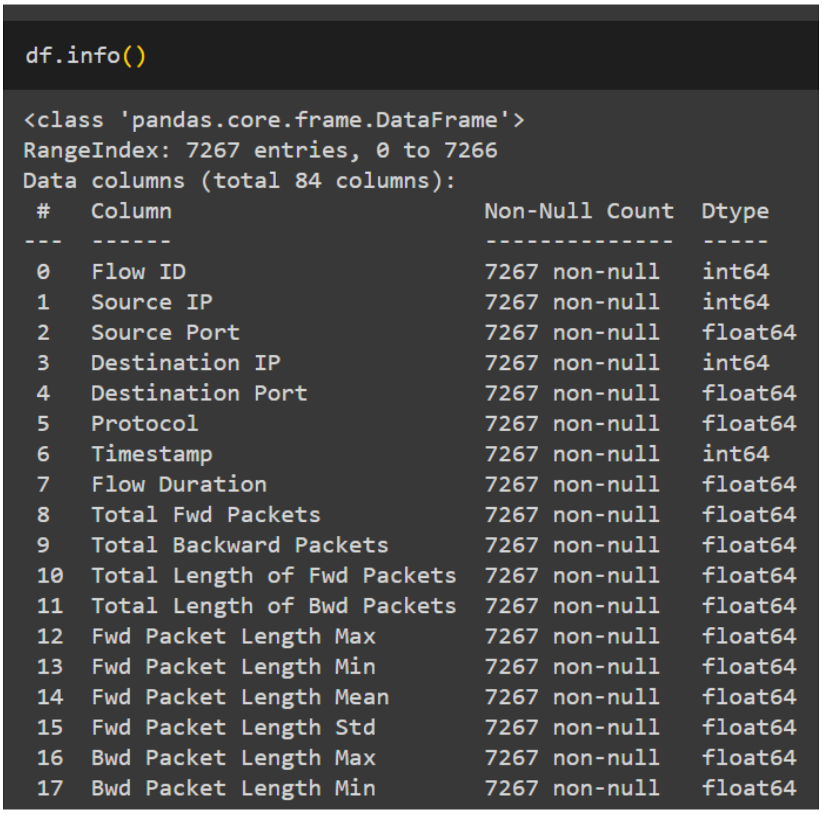
Table 6 - Quantitative Composition of the Dataset

Quantitative Composition of the Dataset		
№	Record Type	Number of Records
1	BENING	2359087
2	DoS Hulk	231072
3	PortScan	158930
4	DDoS	41835
5	DoS GoldenEye	10293
6	FTP-Patator	7938
7	SSH-Patator	5897
8	DoS slowloris	5796
9	DoS Slowhttptest	5499
10	Bot	1966
11	Infiltration	36
12	Heartbleed	11
13	Web Attack – Brute Force	1507
14	Web Attack – XSS	652
15	Web Attack – SQL Injection	21



Flow ID	Source	Source	Destin	Destin	Protoco	Timest	Flow D	Total F	Total B	Total L	Total L	Fwd Pa	Fwd Pa	Fwd Pa	Fwd Pa	Bwd Pa	Bwd Pa	Bwd Pa	Bwd Pa	Flow B	Flow P	Flow IA	Flow IA	Flow IA	Flow IA	Fwd IA
62015	1261	51885.0	1599	53.0	17.0	181	76978.0	2.0	2.0	78.0	206.0	39.0	39.0	39.0	0.0	103.0	103.0	103.0	0.0	3689.3657	51.96289	25659.33	44436.340	76970.0	4.0	4.0
58525	1261	16641.0	1599	53.0	17.0	181	78120.0	2.0	2.0	78.0	206.0	39.0	39.0	39.0	0.0	103.0	103.0	103.0	0.0	3635.4326	51.203277	26040.0	45096.540	78113.0	3.0	3.0
33764	1256	15954.0	1599	53.0	17.0	181	205.0	2.0	2.0	64.0	158.0	32.0	32.0	32.0	0.0	79.0	79.0	79.0	0.0	1082926.8	19512.195	68.333333	83.530433	163.0	5.0	37.0
33749	1256	15744.0	1599	53.0	17.0	181	169.0	2.0	2.0	102.0	224.0	51.0	51.0	51.0	0.0	112.0	112.0	112.0	0.0	1928994.0	23668.639	56.333333	91.511383	162.0	3.0	3.0
34500	1256	28467.0	1599	53.0	17.0	181	297.0	2.0	2.0	102.0	224.0	51.0	51.0	51.0	0.0	112.0	112.0	112.0	0.0	1097643.0	13468.013	99.0	90.415706	184.0	4.0	4.0
36411	1256	60422.0	1599	53.0	17.0	181	164.0	2.0	2.0	64.0	158.0	32.0	32.0	32.0	0.0	79.0	79.0	79.0	0.0	1353658.5	24390.243	54.666666	88.658520	157.0	1.0	1.0
41742	1258	49412.0	1599	135.0	6.0	182	23241857.	23.0	14.0	2054.0	2128.0	168.0	0.0	89.304347	82.663850	268.0	0.0	152.0	78.540533	179.93398	1.5919554	645607.13	2317541.8	11800000.	3.0	23200000.
75605	1263	88.0	1603	49175.0	6.0	182	2.0	3.0	0.0	18.0	0.0	6.0	6.0	6.0	0.0	0.0	0.0	0.0	0.0	9000000.0	1500000.0	1.0	0.0	1.0	1.0	2.0
71450	1267	52360.0	1599	389.0	17.0	182	114.0	2.0	2.0	414.0	326.0	207.0	207.0	207.0	0.0	163.0	163.0	163.0	0.0	6491228.0	35087.719	38.0	59.757844	107.0	3.0	4.0
69780	1265	58702.0	1599	53.0	17.0	182	165.0	2.0	2.0	102.0	224.0	51.0	51.0	51.0	0.0	112.0	112.0	112.0	0.0	1975757.5	24242.424	55.0	55.244909	113.0	3.0	3.0
49745	1258	49438.0	1599	88.0	6.0	182	501.0	7.0	4.0	2828.0	2868.0	1405.0	0.0	404.0	683.81844	1434.0	0.0	717.0	827.92028	11400000.	21956.087	50.1	88.358424	296.0	1.0	501.0
75608	1267	49181.0	1599	88.0	6.0	182	943.0	9.0	6.0	3108.0	3004.0	1539.0	0.0	345.33333	676.75013	1496.0	0.0	500.66666	770.98655	6481442.2	15906.680	67.357142	185.38675	708.0	1.0	943.0
71459	1263	445.0	1603	49155.0	6.0	182	65.0	3.0	2.0	18.0	12.0	6.0	6.0	6.0	0.0	6.0	6.0	6.0	0.0	461538.46	76923.076	16.25	22.май	49.0	1.0	51.0
75683	1268	1029.0	1599	49671.0	6.0	182	26261439.	21.0	14.0	3720.0	2672.0	936.0	0.0	177.14285	268.13677	952.0	0.0	190.85714	324.95937	243.39865	1.3327525	772395.26	4284719.6	25000000.	1.0	26300000.

Figure 10 – A Fragment of the Dataset in Tabular Format



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7267 entries, 0 to 7266
Data columns (total 84 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   Flow ID                      7267 non-null   int64
 1   Source IP                    7267 non-null   int64
 2   Source Port                  7267 non-null   float64
 3   Destination IP               7267 non-null   int64
 4   Destination Port             7267 non-null   float64
 5   Protocol                     7267 non-null   float64
 6   Timestamp                    7267 non-null   int64
 7   Flow Duration                7267 non-null   float64
 8   Total Fwd Packets            7267 non-null   float64
 9   Total Backward Packets       7267 non-null   float64
 10  Total Length of Fwd Packets  7267 non-null   float64
 11  Total Length of Bwd Packets  7267 non-null   float64
 12  Fwd Packet Length Max        7267 non-null   float64
 13  Fwd Packet Length Min        7267 non-null   float64
 14  Fwd Packet Length Mean       7267 non-null   float64
 15  Fwd Packet Length Std        7267 non-null   float64
 16  Bwd Packet Length Max        7267 non-null   float64
 17  Bwd Packet Length Min        7267 non-null   float64
```

Figure 11 – A Fragment of the Dataset Loaded in Google Colab

Complete and well-prepared data are essential for building a good classifier. In reviews of the CICIDS2017 dataset (Intrusion2017, Panigrahi2018, Sharafaldin2018), some researchers noted issues with class imbalance, complex file structure, and missing values. These aspects can generally be considered non-critical.

2.2.2 Data Preprocessing

It is important to note that in the study by Kahraman Kostas, "Anomaly Detection in Networks Using Machine Learning," discrepancies in results were found by other authors when using the selected CICIDS2017 dataset.

To reduce computation time, a subset named "WebAttacks" with a single class of attacks (web attacks - Brute Force, XSS, SQL Injection) was used in the training set based on processing the file Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv from the CICIDS2017 dataset. The WebAttacks set includes 458,968 records, of which 2,180 pertain to web attacks, while the rest pertain to normal traffic.

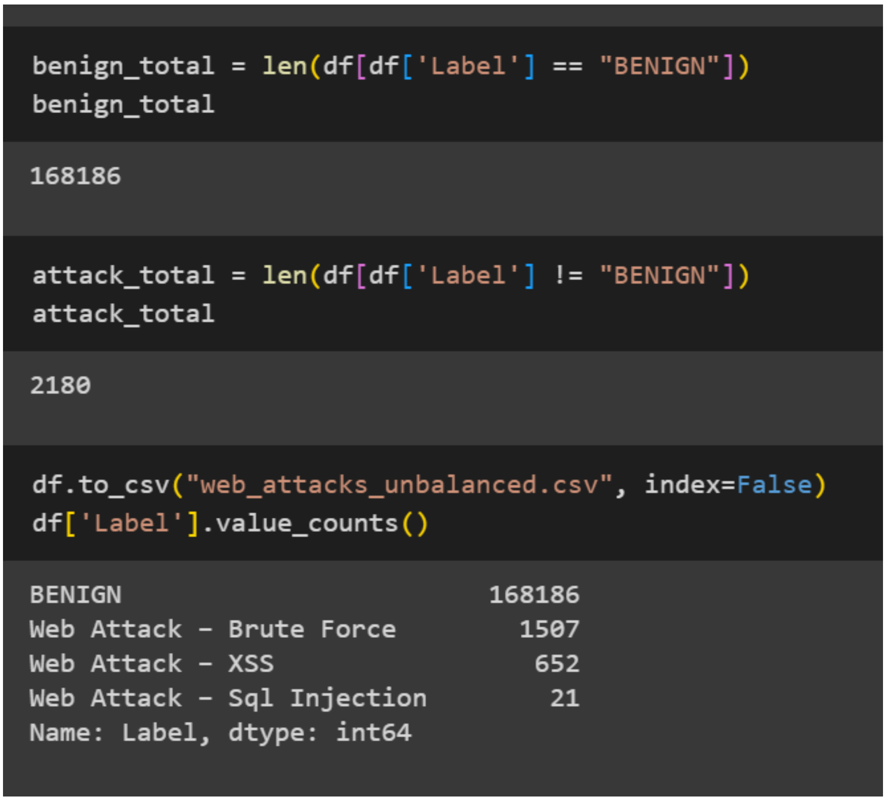
This decision simplifies the task and reduces the quality of the final conclusions - the multiclass classification was reduced to binary, and the size of the training set was reduced.

This subset is publicly available in the repository and was prepared through the following data preprocessing steps:

1. Exclusion of the "Fwd Header Length.1" feature (the "Fwd Header Length" and "Fwd Header Length.1" features are identical).
2. Removal of records with null values in the "Flow ID" session identifier (out of 458,968 records, 170,366 records remained after removal).
3. Replacement of non-numeric values of the "Flow Bytes/s" and "Flow Packets/s" features with -1.
4. Replacement of undefined (NaN) and infinite values with -1.
5. Conversion of string values of the "Flow ID," "Source IP," "Destination IP," and "Timestamp" features to numeric values using label encoding.
6. Encoding of responses in the training set according to the rule: 0 - "no attack," 1 - "attack present."

2.2.3 Sampling Against Class Imbalance

The prepared "WebAttacks" subset is imbalanced: out of a total of 170,366 records, the "no attack" class constitutes 168,186 records, while the "attack present" class constitutes 2,180 records (Figure 12).



```
benign_total = len(df[df['Label'] == "BENIGN"])
benign_total
```

```
168186
```

```
attack_total = len(df[df['Label'] != "BENIGN"])
attack_total
```

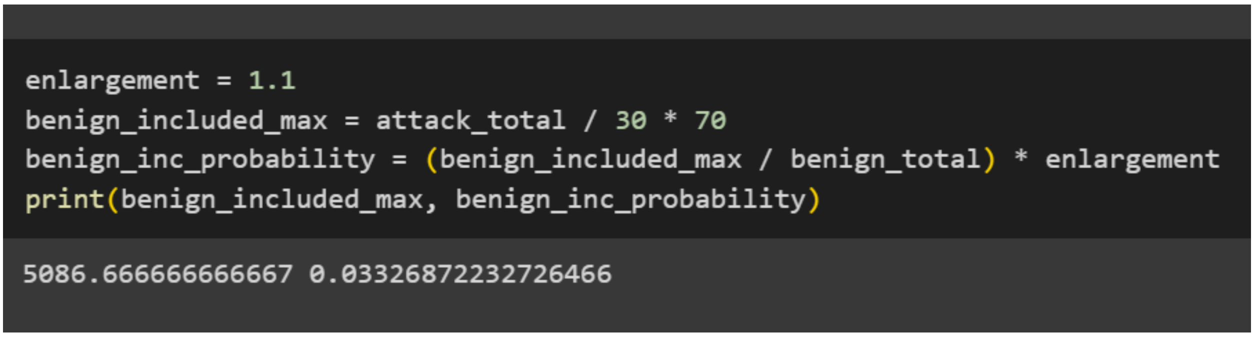
```
2180
```

```
df.to_csv("web_attacks_unbalanced.csv", index=False)
df['Label'].value_counts()
```

```
BENIGN                          168186
Web Attack – Brute Force          1507
Web Attack – XSS                   652
Web Attack – Sql Injection          21
Name: Label, dtype: int64
```

Figure 12– Number of Records in the Imbalanced "WebAttacks" Subset.

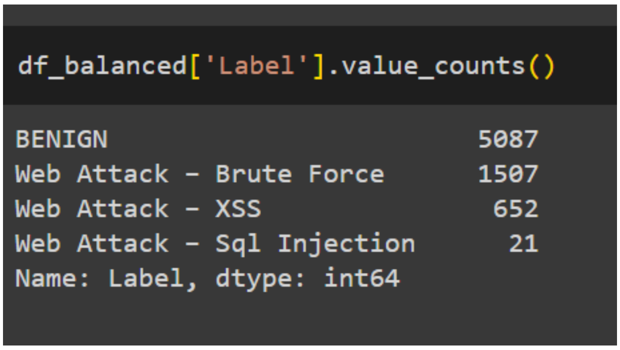
To address the class imbalance, the random sampling method (undersampling) was employed, which involves removing randomly selected instances of the "BENIGN" class. The target ratio of the number of instances between the "BENIGN" class and the "ATTACK" class is 70% (5087 records) / 30% (2180 records).



```
enlargement = 1.1
benign_included_max = attack_total / 30 * 70
benign_inc_probability = (benign_included_max / benign_total) * enlargement
print(benign_included_max, benign_inc_probability)
```

```
5086.666666666667 0.03326872232726466
```

Figure 13 - Formation of the balanced dataset df_balanced.



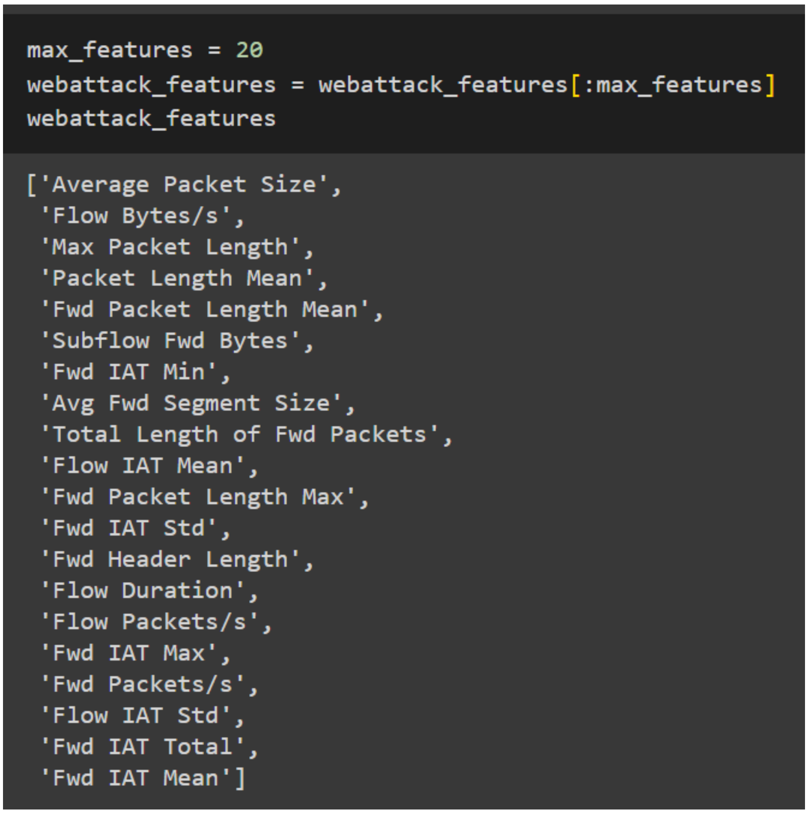
```
df_balanced['Label'].value_counts()
```

```
BENIGN                           5087
Web Attack – Brute Force         1507
Web Attack – XSS                  652
Web Attack – Sql Injection         21
Name: Label, dtype: int64
```

Figure 14 - Dataset df_balanced.

2.2.4 Feature Space Reduction

Results of feature importance assessment and selection for the dataset we are investigating were found in publicly available repositories, which formed the basis for further research (Figure 15).

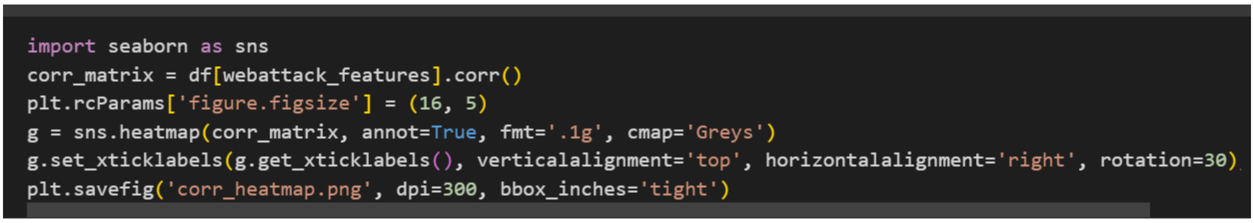


```
max_features = 20
webattack_features = webattack_features[:max_features]
webattack_features
```

```
['Average Packet Size',
 'Flow Bytes/s',
 'Max Packet Length',
 'Packet Length Mean',
 'Fwd Packet Length Mean',
 'Subflow Fwd Bytes',
 'Fwd IAT Min',
 'Avg Fwd Segment Size',
 'Total Length of Fwd Packets',
 'Flow IAT Mean',
 'Fwd Packet Length Max',
 'Fwd IAT Std',
 'Fwd Header Length',
 'Flow Duration',
 'Flow Packets/s',
 'Fwd IAT Max',
 'Fwd Packets/s',
 'Flow IAT Std',
 'Fwd IAT Total',
 'Fwd IAT Mean']
```

Figure 15 - Final results of significance analysis (top 20 features).

For further analysis, a correlation matrix with linear correlation coefficients (Pearson correlation coefficients) calculated for all pairs of the top twenty most significant features was used. It is presented in Figure 16. The color saturation of the fill is proportional to the correlation coefficient value.



```
import seaborn as sns
corr_matrix = df[webattack_features].corr()
plt.rcParams['figure.figsize'] = (16, 5)
g = sns.heatmap(corr_matrix, annot=True, fmt='.1g', cmap='Greys')
g.set_xticklabels(g.get_xticklabels(), verticalalignment='top', horizontalalignment='right', rotation=30)
plt.savefig('corr_heatmap.png', dpi=300, bbox_inches='tight')
```

Figure 16 - Correlation Analysis

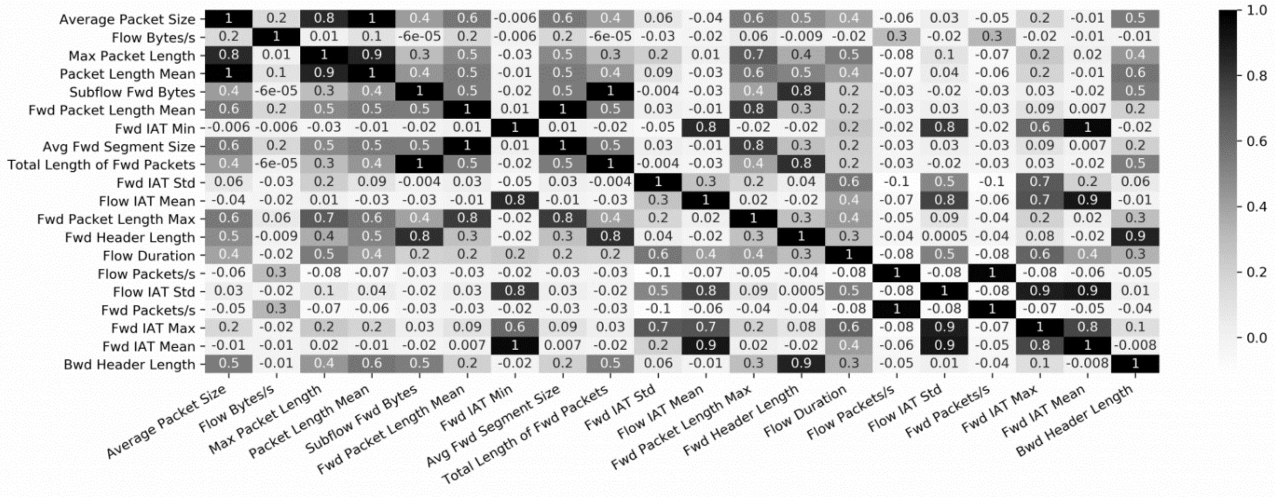


Figure 17 - Results of the Correlation Analysis of the Twenty Most Significant Features

The correlation analysis revealed a strong dependence between pairs of features (unnecessary features for training can be excluded):

1. "Average Packet Size" and "Packet Length Mean".
2. "Subflow Fwd Bytes" and "Total Length of Fwd Packets".
3. "Fwd Packet Length Mean" and "Avg Fwd Segment Size".
4. "Flow Duration" and "Fwd IAT Total".
5. "Flow Packets/s" and "Fwd Packets/s".
6. "Flow IAT Max" and "Fwd IAT Max".

2.2.5 Model Selection

At this stage, a comparison of the previously selected 10 most common machine learning models was conducted. Let's consider the models identified during the literature review and frequently used by various researchers.

The quality of classifier responses (models) was compared using the following metrics:

1. Accuracy
2. Precision
3. Recall

4. F1-score

The evaluation of model performance was conducted on the balanced and preprocessed subset of web attacks (WebAttacks) from the CICIDS2017 dataset (with a ratio of normal to abnormal traffic of 70% / 30%, using 20 most significant features) using two approaches - with stratification and cross-validation, and without stratification but with cross-validation. The evaluation results are presented below in Tables 7 and 8.

Table 7 - Evaluation of model performance on the balanced and preprocessed subset of web attacks (WebAttacks) from the CICIDS2017 dataset.

Without stratification train-test split					
Model	**Acc**	**Pr**	**Recall**	**F1**	**Execution**
KNN	0.966	0.936	0.951	0.969	1.02 s
SVM	0.703	0.618	0.032	0.603	46.60 s
CART	0.965	0.928	0.957	0.964	0.54 s
RF	0.968	0.959	0.911	0.963	0.42 s
ABoost	0.974	0.966	0.947	0.971	8.02 s
LR	0.956	0.970	0.880	0.947	3.14 s
NB	0.735	0.532	0.990	0.775	0.20 s
LDA	0.933	0.909	0.862	0.940	0.65 s
QDA	0.866	0.706	0.656	0.866	0.21 s
MLP	0.942	0.874	0.920	0.964	77.35 s

Table 8 - Evaluation of model performance on the balanced and preprocessed subset of web attacks (WebAttacks) from the CICIDS2017 dataset.

With stratification on train-test split					
Model	**Acc**	**Pr**	**Recall**	**F1**	**Execution**
KNN	0.971	0.943	0.957	0.968	1.38 s
SVM	0.702	0.558	0.025	0.603	33.90 s
CART	0.971	0.953	0.953	0.964	0.68 s
RF	0.971	0.974	0.938	0.965	0.63 s
ABoost	0.970	0.976	0.947	0.971	11.28 s
LR	0.961	0.969	0.898	0.952	3.23 s
NB	0.736	0.532	0.990	0.775	0.18 s
LDA	0.939	0.915	0.879	0.940	0.94 s
QDA	0.924	0.935	0.814	0.949	0.30 s
MLP	0.948	0.907	0.923	0.941	21.77 s

Так, as expected, the models (algorithms) KNN, CART, RF, AdaBoost, and LR demonstrated the best results. The most optimal model considering the combination of the above parameters is the RandomForestClassifier (RF).

It's worth noting that this algorithm, due to quasi-optimal hyperparameter tuning, showed different results for some researchers: Kahraman Kostas' study resulted in Recall 0.94 and F1-score 0.94, while the authors of the CICIDS2017 dataset reported Recall 0.97 and F1-score 0.97.

For intrusion detection algorithms, it's not common to use XGBoost and XGBoost with Principal Component Analysis (PCA).

XGBoost is a machine learning algorithm based on decision tree and gradient boosting framework. It was developed as a research project at the University of Washington. Tianqi Chen and Carlos Guestrin presented their work at the SIGKDD conference in 2016, making a significant impact in the machine learning community. Since its introduction, this algorithm has not only been leading in Kaggle competitions but has also been the foundation of several industry-leading applications. This has led to the formation of a community of data analysis experts contributing to XGBoost open-source projects, with approximately 350 contributors and 3,600 commits on GitHub.

Features of the framework include:

1. Wide applicability: it can be used for regression, classification, ranking, and custom prediction tasks.

2. Compatibility: Works on Windows, Linux, and OS X.

3. Language support: Supports most major programming languages such as C++, Python, R, Java, Scala, and Julia.

4. Cloud integration: Supports AWS, Azure, and Yarn clusters, and integrates well with Flink and Spark.

XGBoost is based on the gradient boosting method of decision trees, which is used for classification and regression tasks. This method creates a prediction model as an ensemble of weak models, usually decision trees. Training occurs sequentially, where each new model predicts the deviations of the previous ensemble on the training set. By adding the predictions of a new tree to the predictions of the trained ensemble, the average deviation of the model can be reduced, which is the target of the optimization problem. Adding new trees allows reducing the model's error until the "early stopping" criteria are met (forms of regularization used to prevent overfitting when training the model with an iterative method like gradient descent; with such methods, the model is updated after each iteration to better fit the training data, and up to a certain point, this also improves the model's performance on data not in the training set, but after that point, the improvement in fitting the training data occurs at the expense of increasing generalization error).

Let's consider a visual illustration of boosting in Figure 18. It shows the behavior of the model at a single point in an abstract linear regression task. Suppose the first model of the ensemble, F, always outputs the sample mean of the predicted value, f0. Such a prediction is quite rough, so the mean squared deviation at the selected point will be quite large. To correct this, we train a model Δ1, which will "adjust" the prediction of the previous ensemble F0. Thus, we obtain an ensemble F1, the prediction of which will be the sum of the predictions of models f0 and Δ1. Continuing this sequence, we arrive at an ensemble F4, the prediction of which is the sum of predictions f0, Δ1, Δ2, Δ3, Δ4, and precisely predicts the value of the given target.

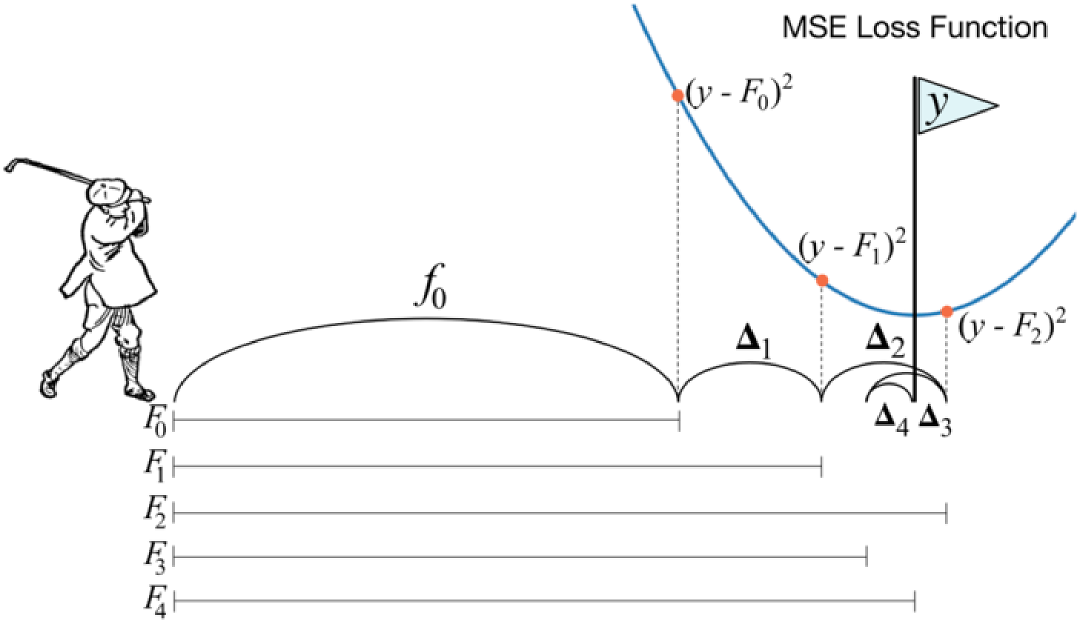


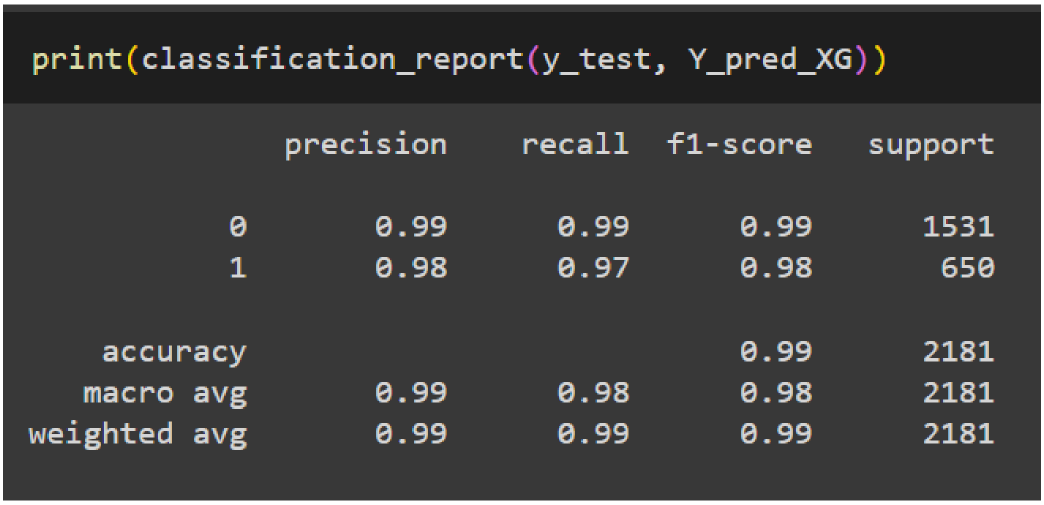
Figure 18 - Boosting Illustration.

XGBoost supports integration with libraries such as scikit-learn, offering regularization capabilities. It supports three main forms of gradient boosting:

- Standard gradient boosting with the ability to adjust the learning rate.
- Stochastic gradient boosting with the ability to sample rows and columns of the dataset.
- Regularized gradient boosting with L1 and L2 regularization.

The implementation of the algorithm is designed for efficiency in computational resources such as time and memory. The project's goal was to maximize the utilization of available resources for model training. Some key implementation features of the algorithm include various strategies for handling missing data, a block structure to support parallelization of tree training, and support for continuing training to fine-tune on new data.

Let's examine the behavior of the XGBClassifier algorithm from the XGBoost library. When configuring a classification model in Python, you can use the `classification_report()` function from the sklearn library to generate three performance metrics for the algorithm (see Figure 19).

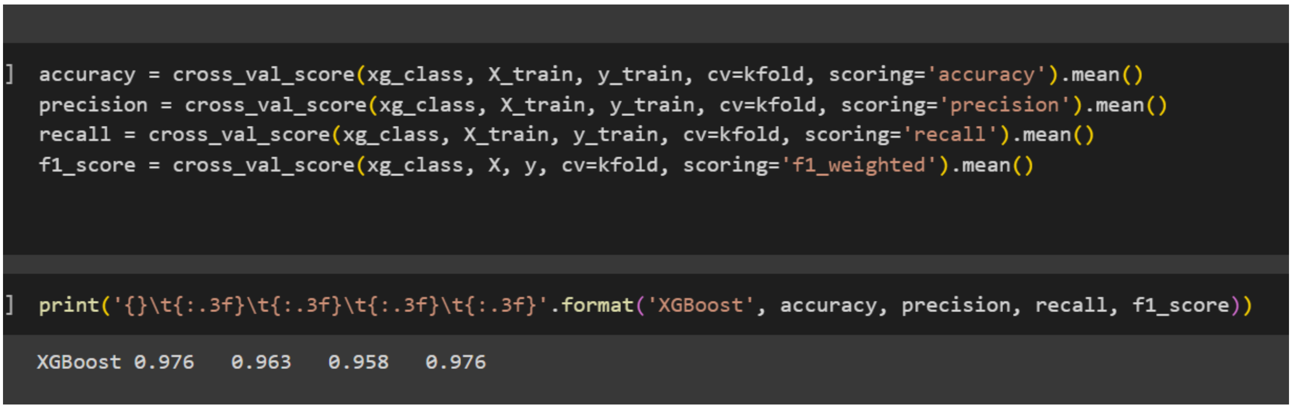


```
print(classification_report(y_test, Y_pred_XG))
```

```
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1531
           1       0.98      0.97      0.98       650

    accuracy                           0.99      2181
   macro avg       0.99      0.98      0.98      2181
weighted avg       0.99      0.99      0.99      2181
```

Figure 19 - classification_report() for XGBClassifier.

Let's highlight the final key indicators of the algorithm (Figure 20):

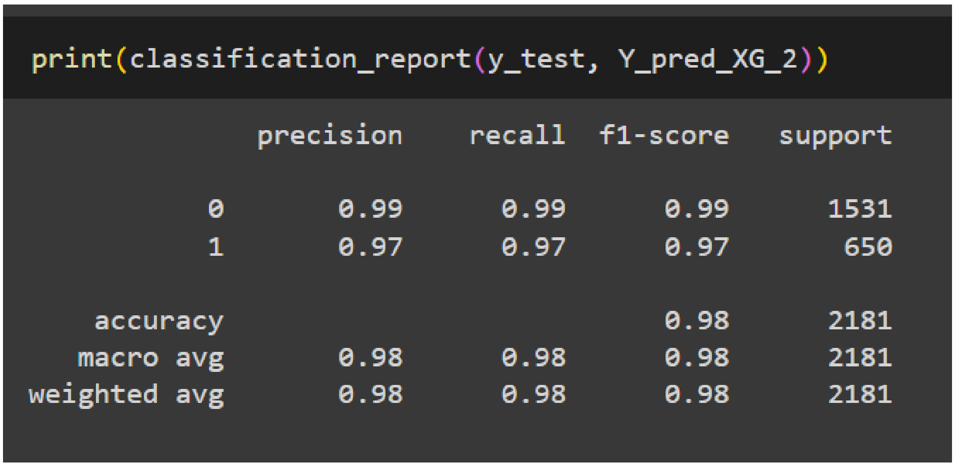


```
]  accuracy = cross_val_score(xg_class, X_train, y_train, cv=kfold, scoring='accuracy').mean()
   precision = cross_val_score(xg_class, X_train, y_train, cv=kfold, scoring='precision').mean()
   recall = cross_val_score(xg_class, X_train, y_train, cv=kfold, scoring='recall').mean()
   f1_score = cross_val_score(xg_class, X, y, cv=kfold, scoring='f1_weighted').mean()
```

```
]  print('{}\t{:.3f}\t{:.3f}\t{:.3f}\t{:.3f}'.format('XGBoost', accuracy, precision, recall, f1_score))

   XGBoost 0.976   0.963   0.958   0.976
```

Figure 20 - Key indicators of the XGBClassifier algorithm.

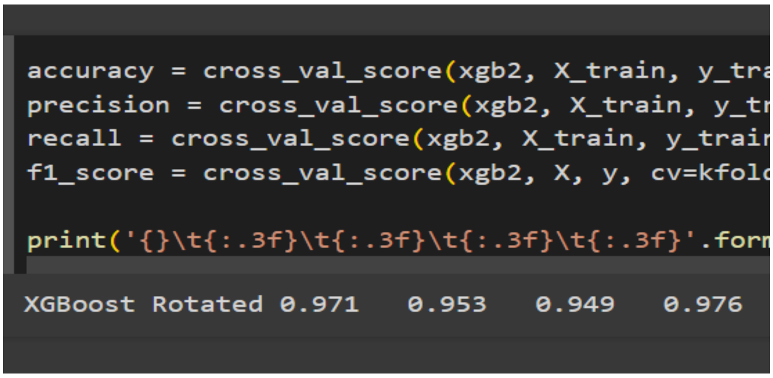
It is also proposed to consider the behavior of the XGBClassifier algorithm using Principal Component Analysis (PCA) for signal decomposition into components, using the PCA module from the sklearn.decomposition package. The performance indicators of the algorithm are shown below in Figure 21.



```
print(classification_report(y_test, Y_pred_XG_2))

              precision    recall  f1-score   support

           0       0.99      0.99      0.99      1531
           1       0.97      0.97      0.97       650

    accuracy                           0.98      2181
   macro avg       0.98      0.98      0.98      2181
weighted avg       0.98      0.98      0.98      2181
```

Figure 21 - classification_report() for XGBClassifier rotated.

Let's highlight the final key performance indicators of the above-mentioned algorithm (Figure 22):



```
accuracy = cross_val_score(xgb2, X_train, y_tra
precision = cross_val_score(xgb2, X_train, y_tr
recall = cross_val_score(xgb2, X_train, y_trair
f1_score = cross_val_score(xgb2, X, y, cv=kfolc

print('{}\t{:.3f}\t{:.3f}\t{:.3f}\t{:.3f}'.forn
```

```
XGBoost Rotated 0.971    0.953    0.949    0.976
```

Figure 22 - Key Performance Indicators of the XGBClassifier Rotated Algorithm.

Also, an evaluation of the algorithms was conducted using stratification. To summarize the obtained results of the proposed and previously compared algorithms, let's present them in Table 9.

Table 9 - Evaluation of Model Performance on the Balanced and Preprocessed Subset of WebAttacks in the CICIDS2017 Dataset.

With stratification on train-test split					
Model	**Acc**	**Pr**	**Recall**	**F1**	**Execution**
KNN	0.971	0.943	0.957	0.968	1.38 s
SVM	0.702	0.558	0.025	0.603	33.90 s
CART	0.971	0.953	0.953	0.964	0.68 s
RF	0.971	0.974	0.938	0.965	0.63 s
ABoost	0.970	0.976	0.947	0.971	11.28 s
LR	0.961	0.969	0.898	0.952	3.23 s
NB	0.736	0.532	0.990	0.775	0.18 s
LDA	0.939	0.915	0.879	0.940	0.94 s
QDA	0.924	0.935	0.814	0.949	0.30 s
MLP	0.948	0.907	0.923	0.941	21.77 s
XGBoost	0.976	0.971	0.963	0.976	
XGBoost Rotated	0.971	0.957	0.957	0.976	

Practical results have been obtained regarding the performance of 12 models, from which the best ones need to be selected.

In this study, a multi-criteria evaluation matrix is used, where each parameter is assigned a specific "weight", and models are rated on a scale from 1 to 3, where 1 - unsatisfactory, 2 - satisfactory, and 3 - good. This method minimizes the likelihood of error and provides a clear assessment of priority mathematics. Parameters such as algorithm accuracy (Accuracy) and precision (Precision) are assigned a weight of 0.3, while the others are assigned 0.2 each.

Thanks to this method, we can consider multiple selection parameters present in the study to evaluate the models. Table 10 shows the final evaluation results of the models, from which it can be concluded that the Adaptive Boosting over Decision Tree (AdaBoost, sklearn.ensemble.AdaBoostClassifier) and XGBoost gradient boosting algorithms can be considered suitable for solving the stated tasks.

Table 10 - Multi-criteria evaluation of models.

Model	Acc	Pr	Recall	F1	Result
	0,3	0,3	0,2	0,2	max=3
KNN	3	1	3	2	2,2
SVM	1	2	1	1	1,3
CART	3	2	3	2	2,5
RF	3	3	3	2	2,8
ABoost	3	3	2	2	2,6
LR	2	2	1	2	1,8
NB	1	1	3	1	1,4
LDA	2	1	1	1	1,3
QDA	2	1	1	1	1,3
MLP	2	1	2	1	1,5
XGBoost	3	3	3	3	3
XGBoost Rotated	3	2	3	3	2,7

According to the obtained results, RF and XGBoost gradient boosting algorithm can be considered suitable for solving the stated tasks.

Considering the comparison conducted, the authors of the study suggest considering two types of algorithms:

1. Combining neural networks with traditional machine learning methods, such as RandomForestClassifier (using neural networks for feature selection together with RandomForestClassifier for classification).
2. VotingClassifier model containing RandomForestClassifier and XGBClassifier (an intrusion detection system model based on the VotingClassifier ensemble model containing random forest and XGBClassifier models).

Let's consider each of them in more detail.

Combining a neural network with a Random Forest classifier for tabular data classification can be useful for extracting complex features using a neural network and then using these features for a more interpretable or robust classifier, such as Random Forest.

The algorithm of the approach looks as follows:

1. Training a neural network to extract features: train a neural network to extract features from tabular data. Instead of using the network output for classification, take the intermediate layer containing useful data representations (features).
2. Using these features to train Random Forest: use the extracted features to train the Random Forest model.

The RF algorithm itself will work as follows (Figure 23):

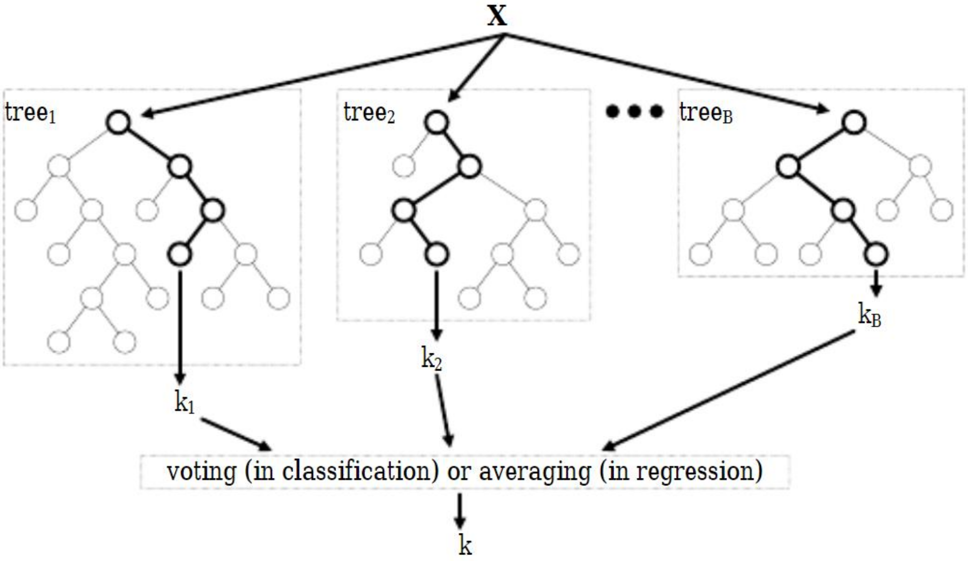


Figure 23 - RF Architecture.

For the classification task, a majority voting solution is chosen, while for regression, it's the average.

This approach combines the advantages of neural networks for extracting complex features and the stability of Random Forests for final classification.

Feature extraction from tabular data is the process of transforming raw data into a format better suited for analysis and modeling. It's a crucial step in machine learning as the right features can significantly improve model performance. In the context of tabular data, feature extraction can involve various techniques such as:

1. Direct use of features. With well-prepared data, original features can be directly used for model training.

2. Creating new features. Generating new features based on existing data. This can include mathematical transformations or aggregations.

3. Feature selection. Choosing the most important features from the dataset using feature selection methods.

4. Applying complex models for automatic feature extraction. Utilizing complex models like neural networks to automatically extract complex features from the data.

When we talk about feature extraction using a neural network, we mean using the intermediate layers of the network to create new data representations. These representations can capture higher-level information than the original features. The intermediate outputs of the network can be used as new features for other models, such as Random Forest.

Now, let's consider the second algorithm – VotingClassifier.

In machine learning, an ensemble of models refers to a combination of several learning algorithms that, when working together, allow for building a more effective and accurate model. The goal of ensemble methods is to combine predictions from multiple base estimators, built with a specified learning algorithm, to improve generalization/reliability compared to a single estimator.

VotingClassifier is a machine learning model that is trained on an ensemble of multiple models and predicts the result (class) based on their highest probability of the selected class as the output. It simply aggregates the results of each classifier passed into the voting classifier and predicts the output class based on the majority of the votes.

The idea is that instead of creating individually conceptually different machine learning classifier models and determining accuracy for each of them, we create a single model that is trained using these models and predicts the output based on their aggregate majority votes for each output class. The aggregate solution often provides better generalization and predictive performance than individual models.

The scikit-learn library provides a convenient implementation of the voting classifier, allowing for easy integration and experimentation with various models in a unified environment. This approach is particularly useful when working with different data patterns and provides more reliable predictions, making it a valuable tool in a machine learning practitioner's toolkit.

Analyzing the behavior of the VotingClassifier model, which contains Random Forest and XGBClassifier, both of which were discussed earlier, is proposed. The architecture of this algorithm is presented in Figure 24.

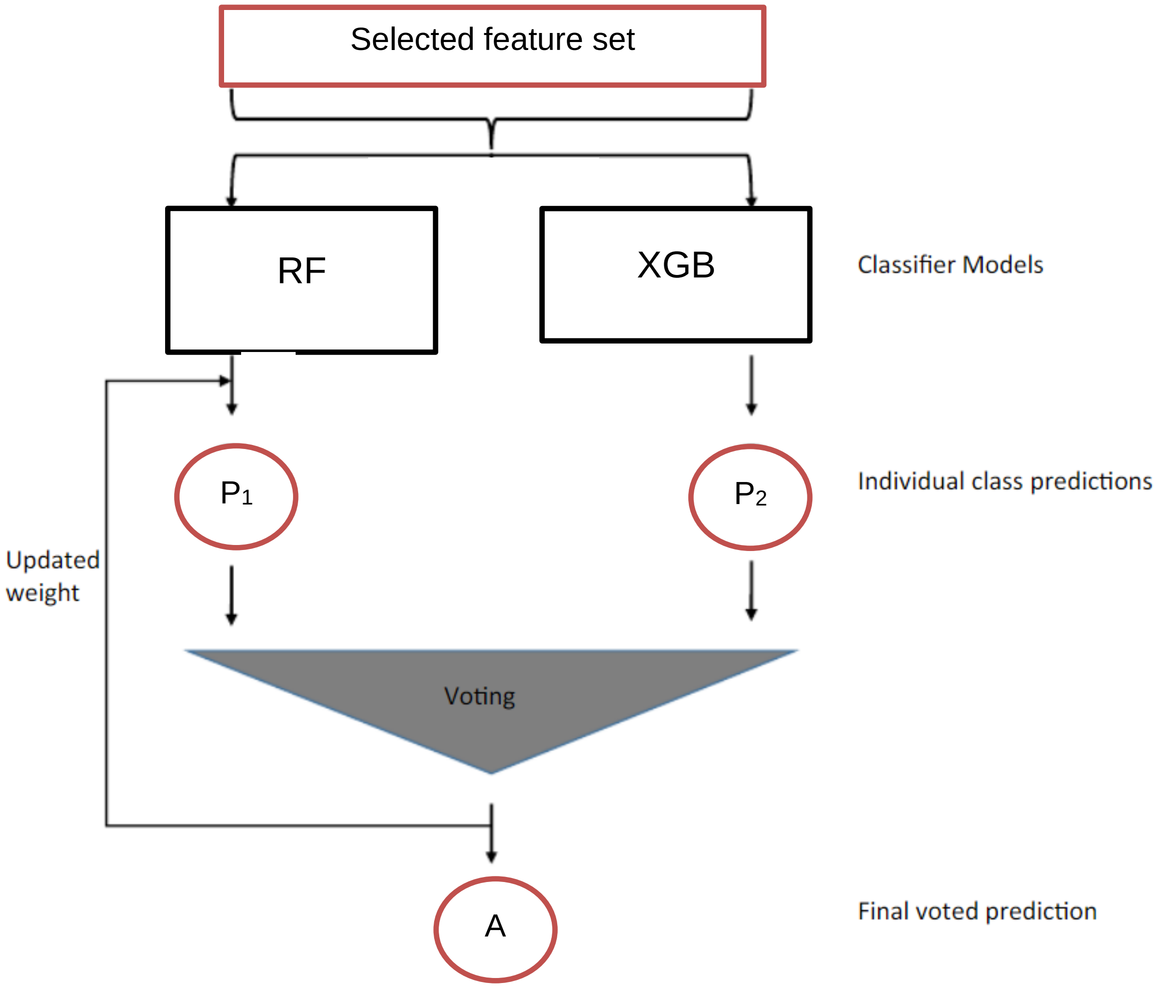
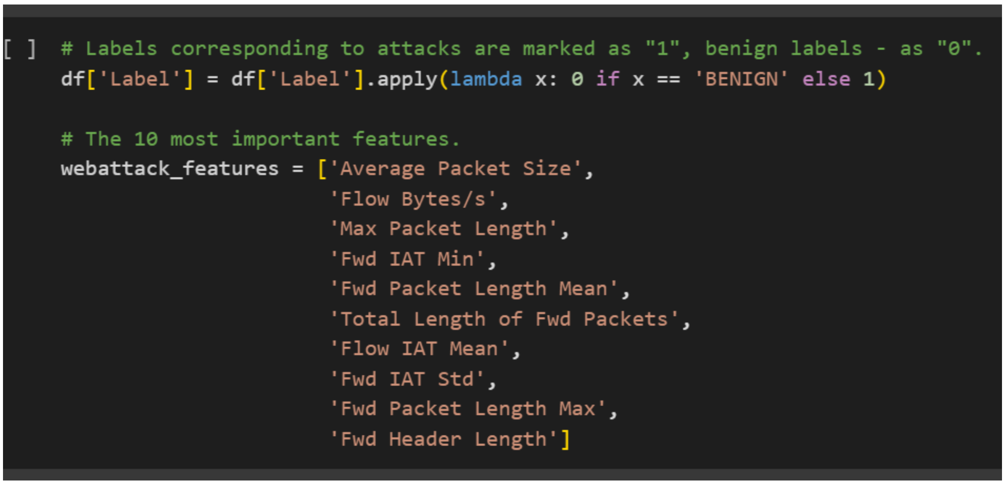


Figure 24 - Architecture of the VotingModel (RF + XGB).

We'll cover some practical aspects of implementing the two proposed methods.

Each algorithm was trained on a balanced and preprocessed subset of WebAttacks web attack data from the CICIDS2017 dataset (with a normal to anomaly traffic ratio of 70% / 30%, using the 10 most significant features selected after the earlier conducted correlation analysis).

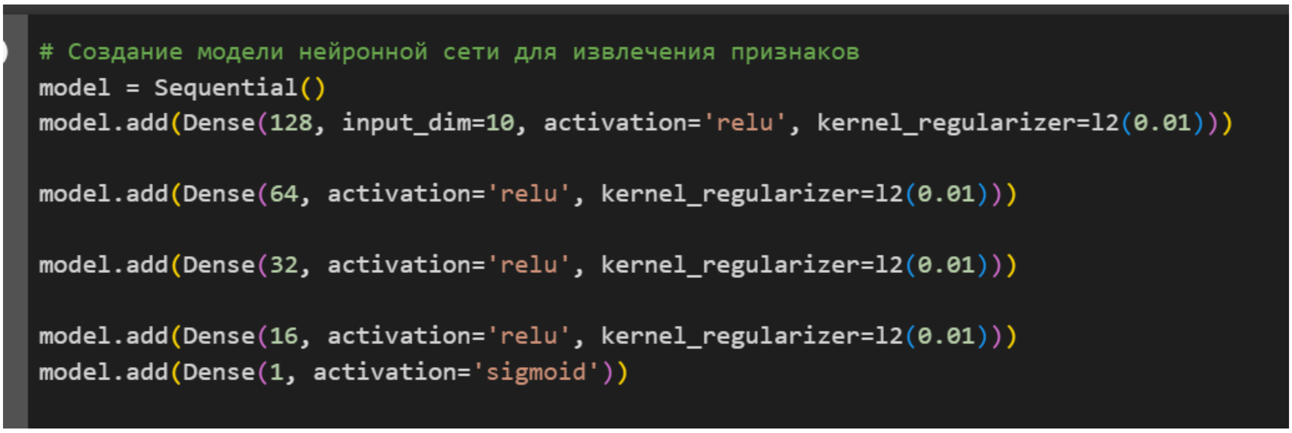
Categorical labels were transformed into numerical form using a simple label encoding: "1" for samples with attacks and "0" for samples without attacks (see Figure 25).



```
[ ]  # Labels corresponding to attacks are marked as "1", benign labels - as "0".
     df['Label'] = df['Label'].apply(lambda x: 0 if x == 'BENIGN' else 1)

     # The 10 most important features.
     webattack_features = ['Average Packet Size',
                           'Flow Bytes/s',
                           'Max Packet Length',
                           'Fwd IAT Min',
                           'Fwd Packet Length Mean',
                           'Total Length of Fwd Packets',
                           'Flow IAT Mean',
                           'Fwd IAT Std',
                           'Fwd Packet Length Max',
                           'Fwd Header Length']
```

Figure 25 - Preparation of features and labels for model training.

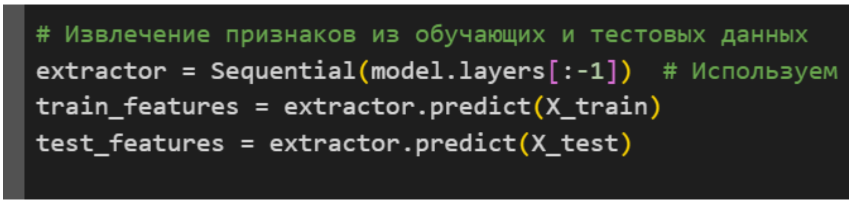
The neural network for feature extraction consists of several fully connected layers with ReLU activation (Figure 26).



```
# Создание модели нейронной сети для извлечения признаков
model = Sequential()
model.add(Dense(128, input_dim=10, activation='relu', kernel_regularizer=l2(0.01)))

model.add(Dense(64, activation='relu', kernel_regularizer=l2(0.01)))

model.add(Dense(32, activation='relu', kernel_regularizer=l2(0.01)))

model.add(Dense(16, activation='relu', kernel_regularizer=l2(0.01)))
model.add(Dense(1, activation='sigmoid'))
```

Figure 26 - Creating a neural network model for feature extraction..

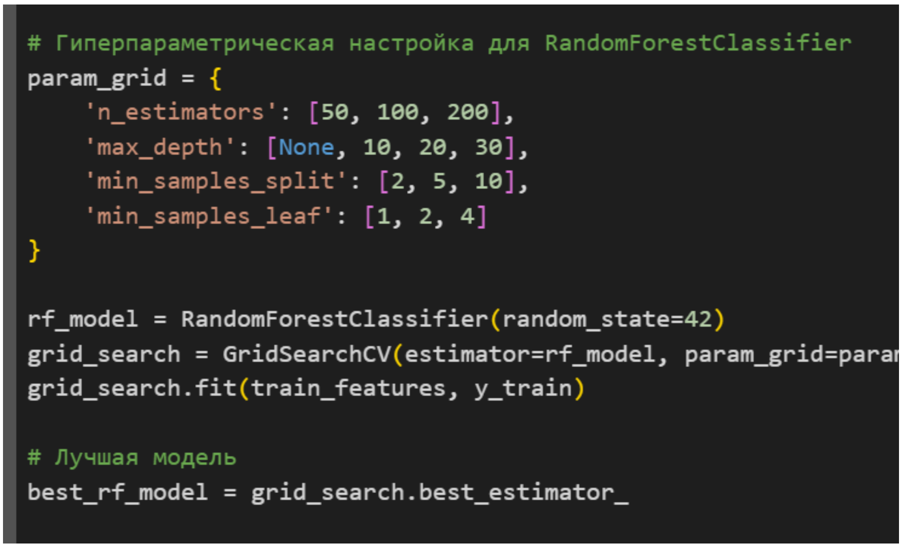
The feature extraction model outputs data from the last hidden layer of the neural network (Figure 27).



```
# Извлечение признаков из обучающих и тестовых данных
extractor = Sequential(model.layers[:-1])  # Используем
train_features = extractor.predict(X_train)
test_features = extractor.predict(X_test)
```

Figure 27 - Feature Extraction.

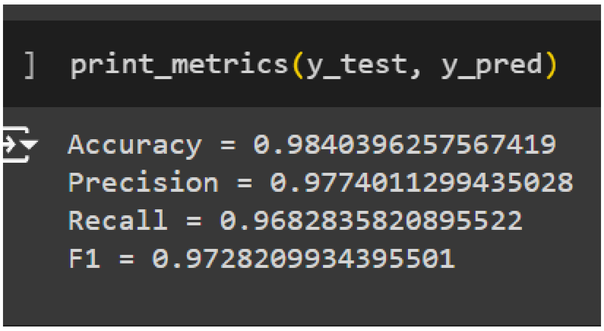
Hyperparameter tuning for the RandomForestClassifier was conducted. GridSearchCV was employed to search for the best hyperparameters of the RandomForestClassifier model. GridSearchCV utilizes cross-validation for a more reliable model evaluation and hyperparameter search, which can help improve the model's performance (Figure 28).



```
# Гиперпараметрическая настройка для RandomForestClassifier
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

rf_model = RandomForestClassifier(random_state=42)
grid_search = GridSearchCV(estimator=rf_model, param_grid=para
grid_search.fit(train_features, y_train)

# Лучшая модель
best_rf_model = grid_search.best_estimator_
```

Figure 28 - Applying GridSearchCV for RF.

Next, the Random Forest model is trained on the extracted features, and the accuracy is evaluated on the test data. The evaluation results are presented below in Figure 29.

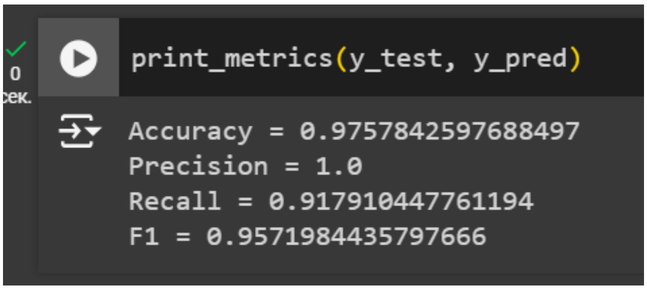


```
print_metrics(y_test, y_pred)
```

```
Accuracy = 0.9840396257567419
Precision = 0.9774011299435028
Recall = 0.9682835820895522
F1 = 0.9728209934395501
```

Figure 29 - Evaluation Metrics of the RF Model with Feature Extraction Neural Network Combination.

For the VotingClassifier model, hyperparameter tuning of the RandomForestClassifier was also conducted using GridSearchCV. The model evaluation results are presented in Figure 30.



```
print_metrics(y_test, y_pred)
```

```
Accuracy = 0.9757842597688497
Precision = 1.0
Recall = 0.917910447761194
F1 = 0.9571984435797666
```

Figure 30 - Evaluation Metrics of the VotingClassifier Model.

Let's summarize the obtained results in tabular form (Table 11).

Table 11 - Evaluation of the Proposed Algorithms.

	VotingClassifier (Random Forest + XGBClassifier)	RF + neural network
Metrics of Model Performance	Values of Metrics	
Confusion_matrix	array([[1280, 1], [44, 492]])	array([[1269, 12], [17, 519]])
Accuracy	0.976	0.988
Precision	1.0	0.981
Recall	0.918	0.978
F1	0.957	0.979

Results and Conclusions of Chapter Two

Thus, an approach to classifying network anomalies has been proposed, their main features have been identified, and the main methods of their detection have been structured. Intrusion detection methods are divided into four major groups: behavioral methods, machine learning methods, computational intelligence methods, and knowledge-based methods.

The general principles of operation of each group of methods have been carefully described, along with their advantages, disadvantages, and better utilization options for each method.

In practical comparisons of known machine learning algorithms used in IDS/IPS, the best results were expectedly demonstrated by models (algorithms) RF, XGBoost, and XGBoost Rotated.

The most optimal combination of the above parameters is the RandomForestClassifier (RF). For training intrusion detection systems among available public datasets, one of the relevant and comprehensive datasets - "Intrusion Detection Evaluation Dataset" CICIDS2017 was chosen, which contains safe and modern common attacks resembling real-world data.

The performance metrics of the XGBClassifier and XGBClassifier with the application of principal component analysis (PCA) were also evaluated.

According to the obtained results, the gradient boosting algorithm (a machine learning algorithm based on decision tree and utilizing the gradient boosting framework) can be considered a suitable algorithm for solving the posed tasks.

Two algorithms were proposed - VotingClassifier (Random Forest + XGBClassifier) and RF + neural network for feature extraction. The second approach showed a higher accuracy metric (0.988), while the first one showed a precision metric (1.0) when trained on the same balanced and preprocessed subset of web attack data (WebAttacks dataset of CICIDS2017) with a normal to anomalous traffic ratio of 70% / 30%, utilizing 10 most significant features selected after a previously conducted correlation analysis.

3 APPLICATION OF ADVERSARIAL LEARNING IN INTRUSION DETECTION ALGORITHMS

3.1 Adversarial Machine Learning

Machine learning systems can become targets of malicious attacks just like vulnerabilities in a firewall can be exploited to gain access to a web server. Therefore, before implementing such systems in the realm of security, it is necessary to carefully examine their weaknesses and understand how susceptible they are to attacks. Adversarial machine learning is the study of vulnerabilities in machine learning systems in hostile environments. Many researchers in the fields of security and machine learning have demonstrated research results on various attacks against antivirus programs [28], spam filters [29], and so forth. Developers of machine learning systems are responsible for preventing attacks and creating means of protection in case of threats to data confidentiality, national security, and human lives.

Some researchers still realize that modern AI-driven security solutions are significantly underdeveloped and have defects [30].

The implementation of the concept of adversarial machine learning is difficult because most machine learning models operate as black boxes. This means that users and specialists cannot precisely understand how models make their predictions due to the lack of transparency in the internal processes of detectors and classifiers. Without explanations about the decisions made, it is difficult for people to determine when a system is being influenced by malicious actors. This creates doubts about the reliability of machine learning systems and leads to resistance to their deployment as primary decision-making tools in the field of security.

Machine learning methodologies are typically developed with preliminary assumptions about data stability, feature independence, and low stochasticity (randomness) [31]. Adversaries violate any assumptions made by specialists until they compute the path into the system with the least resistance.

In fact, when an algorithm is trained on training data, it operates with a limited amount of information, which represents only a portion of the entire theoretical space of possible variations. When the model is tested in laboratory conditions or in real-world practice, the test dataset may contain elements that were not present in the training data. These missing elements are referred to as the "adversarial space."

Malicious actors can exploit these areas of adversarial space to deceive machine learning algorithms. However, an even more serious threat arises when adversaries can interfere with the process of training models and invalidate assumptions about the stability of the data used in machine learning. Since statistical learning models rely on the provided data, vulnerabilities in such systems naturally arise due to mismatches in this data. It is important for specialists to ensure that the data used for training accurately reflects the real distribution to the extent possible. At the same time, it is crucial to continuously monitor various attack methods to enable the development of more robust algorithms and systems.

The goal of this section is to assess the stability of the developed model against adversarial attacks - how difficult (or easy) it will be for a malicious actor to "trick" the system. Adversarial attacks ("adversarial" or "hostile") encompass all known attacks on machine learning models that can be implemented both during the model training stage and during its operation.

Let us highlight the main types of adversarial attacks, as visualized in Figure 31:

1) Poisoning attack (poisoning attack), when an attacker affects the training data during the training phase and, for example, adds incorrectly labeled examples, which leads to model errors during the exploitation phase.

2) Membership inference attack, where an attacker attempts to infer a set of training data while violating its privacy (such attacks are especially dangerous for personal data - facial recognition, medical records, financial transactions, etc.).

3) Model extraction attack (model extraction attack), when an attacker, not knowing the target model, tries to “steal features” of the model.

4) Evasion attack, when an attacker selects input data at the exploitation stage so that the model gives an incorrect response.

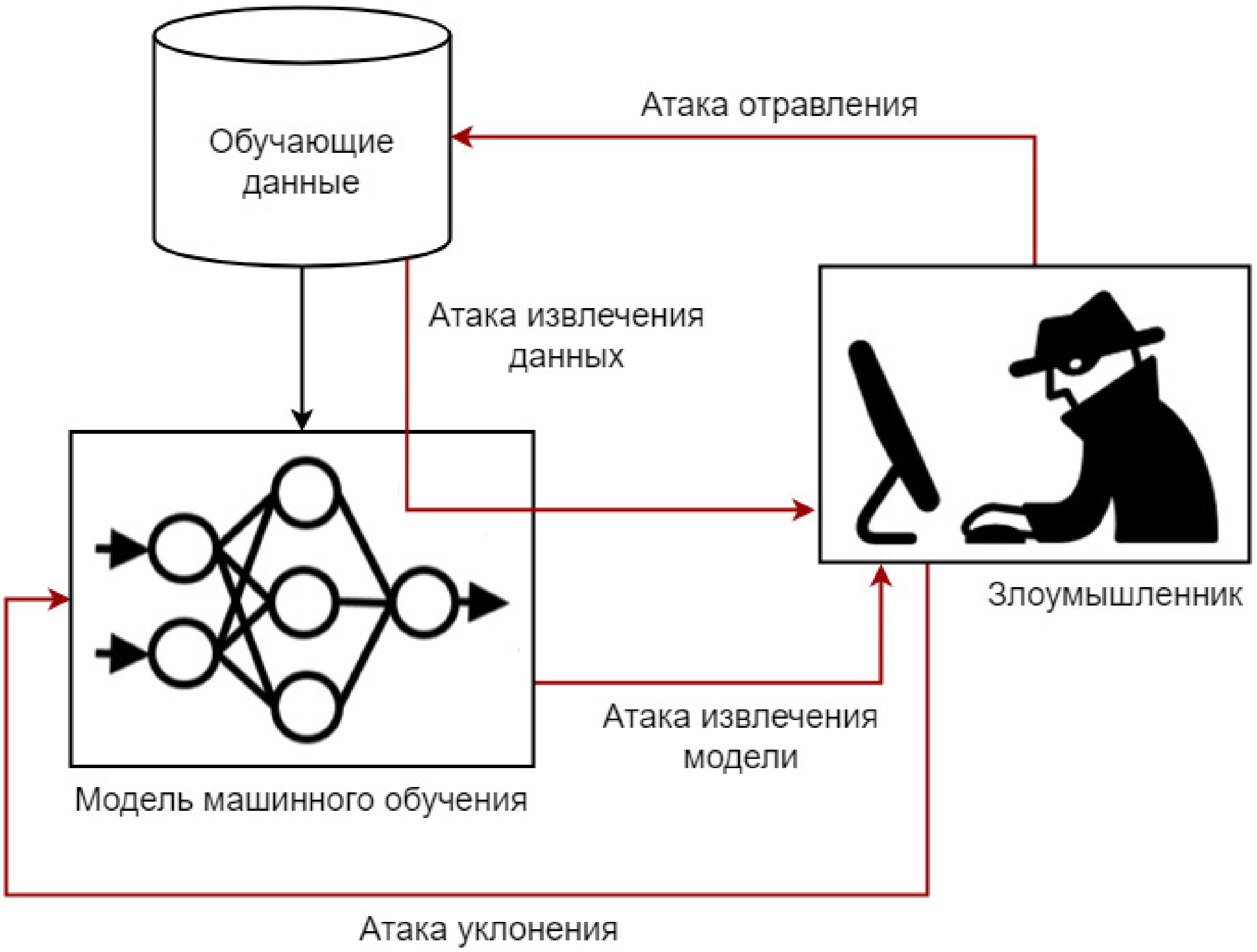
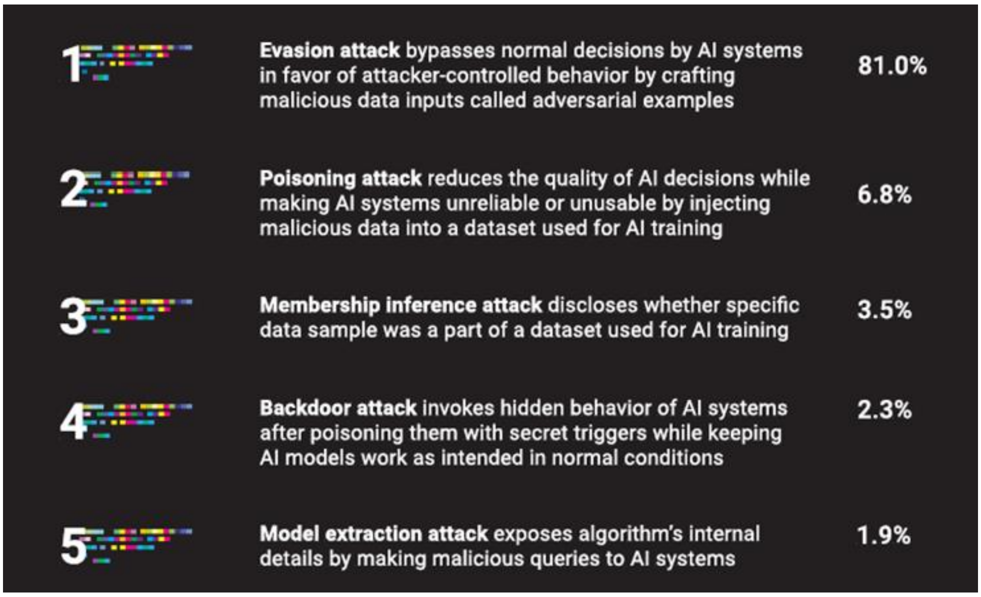


Figure 31 - The main types of adversarial attacks.

According to adversa.ai rankings, one of the most common and easy to understand attacks is the evasion attack (Figure 32). It requires only a basic understanding of the target system, so evasion attacks can be considered one of the most dangerous attacks. In the following we will consider this type of adversarial attack.



1	**Evasion attack** bypasses normal decisions by AI systems in favor of attacker-controlled behavior by crafting malicious data inputs called adversarial examples	81.0%
2	**Poisoning attack** reduces the quality of AI decisions while making AI systems unreliable or unusable by injecting malicious data into a dataset used for AI training	6.8%
3	**Membership inference attack** discloses whether specific data sample was a part of a dataset used for AI training	3.5%
4	**Backdoor attack** invokes hidden behavior of AI systems after poisoning them with secret triggers while keeping AI models work as intended in normal conditions	2.3%
5	**Model extraction attack** exposes algorithm's internal details by making malicious queries to AI systems	1.9%

Figure 32 - Ranking of the most common adversarial attacks according to adversa.ai

Some machine learning models routinely misclassify adversarial examples - input data generated by applying small but intentionally worst-case perturbations to examples in the dataset, so that the distorted input data causes the model to produce an incorrect answer with high confidence. Early attempts to explain this phenomenon focused on nonlinearity and overtraining [32].

In the paper "EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLE" [32], published in 2015 at a conference, showed an example of an evasion attack, where a mask invisible to the human eye is superimposed on an image of a panda, and the recognition model starts to make the mistake of calling the panda a gibbon (Figure 33). The paper also proposed an efficient way to generate adversarial examples, the "Fast Gradient Sign Method" (FGSM). It is this publication that is usually associated with a sharp increase in interest in adversarial attacks.

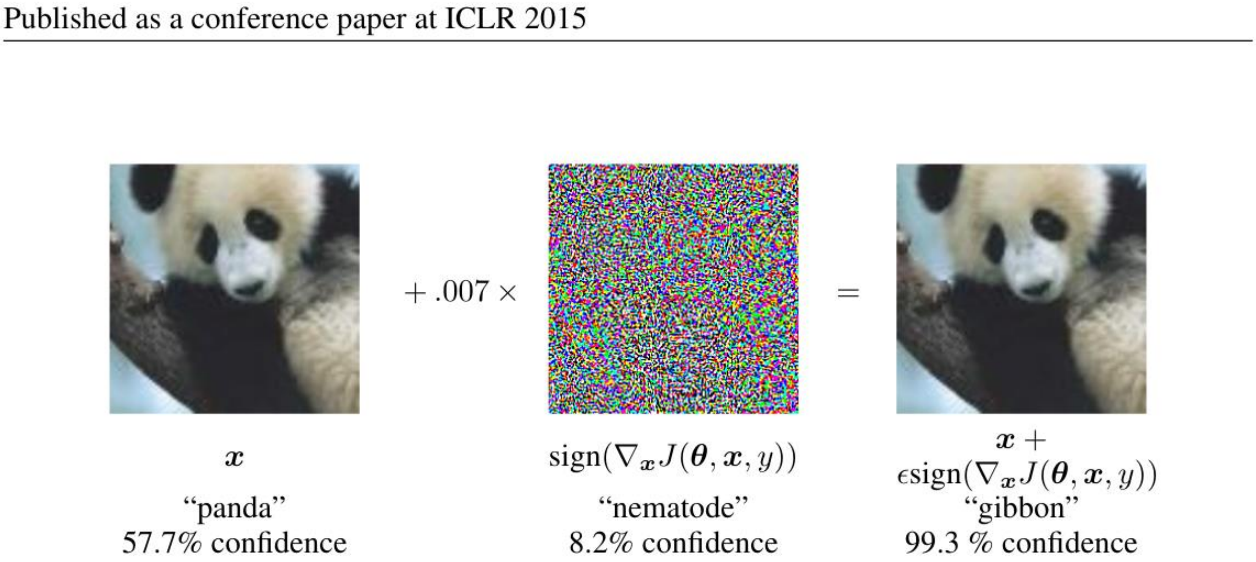

Published as a conference paper at ICLR 2015
+ .007 ×
=
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

Figure 33 - An example of an evasion attack.

Existing algorithms for generating adversarial examples (adversarial attacks) generally involve two steps:

(1) - selecting the direction of the attack. The sensitivity of the model F to changes in the values of individual attributes is evaluated.

(2) - perturbation formation. The obtained knowledge is used to generate a perturbation that affects the classification of the sample X. If the model F misclassifies the result X + δX instead of the original class, it means that an adversarial example has been found. Otherwise, the above steps are repeated, e.g., already for the example X + δX.

The operation of the evasion attack is shown in general in Figure 34.

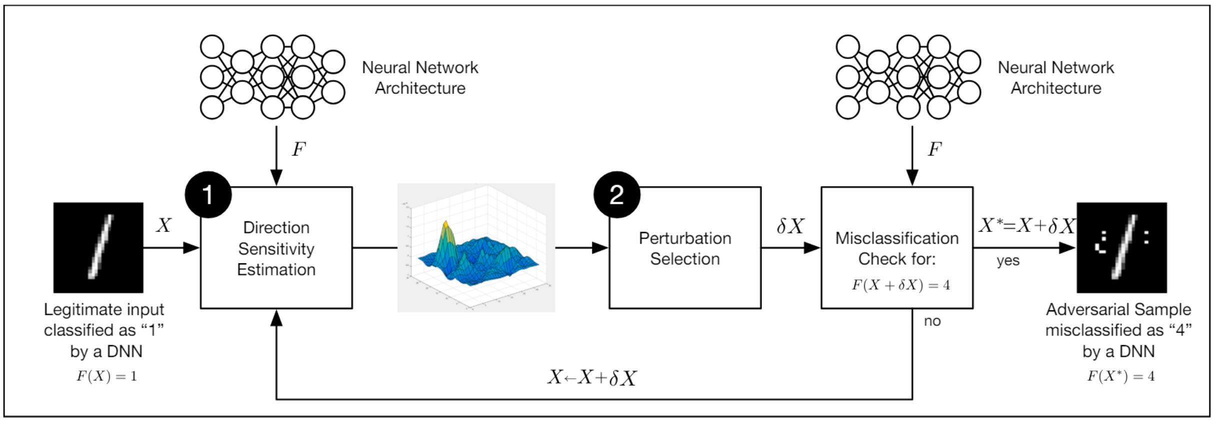
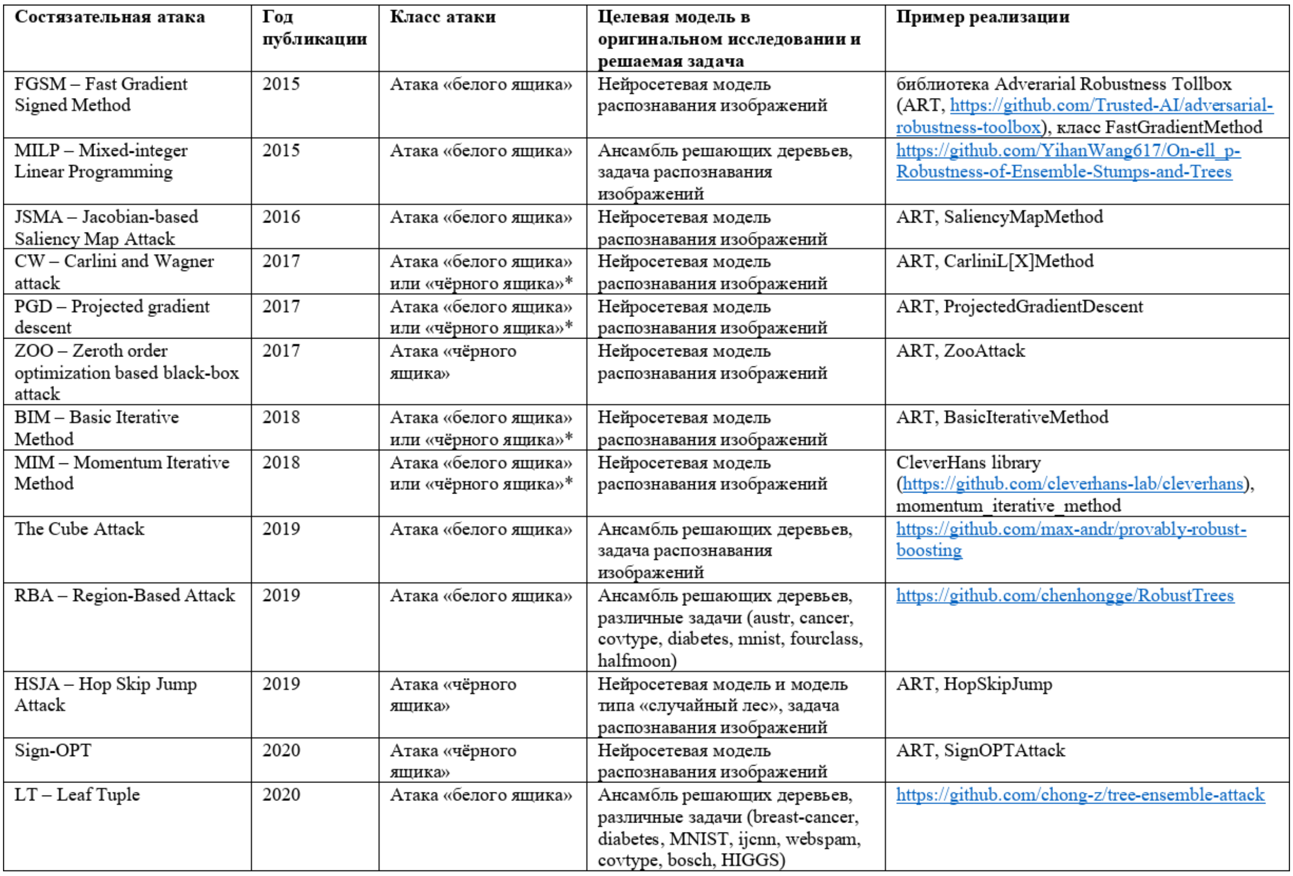

Neural Network Architecture
Neural Network Architecture
F
F
X
1
Direction Sensitivity Estimation
2
Perturbation Selection
δX
Misclassification Check for:
$F(X+\delta X)=4$
$X^{*}=X+\delta X$
yes
no
Legitimate input classified as "1" by a DNN
$F(X)=1$
Adversarial Sample misclassified as "4" by a DNN
$F(X^{*})=4$
$X \leftarrow X+\delta X$

Figure 34 - Evasion attack steps in the general case [32].

Let us turn to the table with adversarial attacks contained in the most cited studies in recent years (Figure 35).



Состязательная атака	Год публикации	Класс атаки	Целевая модель в оригинальном исследовании и решаемая задача	Пример реализации
FGSM – Fast Gradient Signed Method	2015	Атака «белого ящика»	Нейросетевая модель распознавания изображений	библиотека Adverarial Robustness Tollbox (ART, https://github.com/Trusted-AI/adversarial-robustness-toolbox), класс FastGradientMethod
MILP – Mixed-integer Linear Programming	2015	Атака «белого ящика»	Ансамбль решающих деревьев, задача распознавания изображений	https://github.com/YihanWang617/On-ell_p-Robustness-of-Ensemble-Stumps-and-Trees
JSMA – Jacobian-based Saliency Map Attack	2016	Атака «белого ящика»	Нейросетевая модель распознавания изображений	ART, SaliencyMapMethod
CW – Carlini and Wagner attack	2017	Атака «белого ящика» или «чёрного ящика»*	Нейросетевая модель распознавания изображений	ART, CarliniL[X]Method
PGD – Projected gradient descent	2017	Атака «белого ящика» или «чёрного ящика»*	Нейросетевая модель распознавания изображений	ART, ProjectedGradientDescent
ZOO – Zeroth order optimization based black-box attack	2017	Атака «чёрного ящика»	Нейросетевая модель распознавания изображений	ART, ZooAttack
BIM – Basic Iterative Method	2018	Атака «белого ящика» или «чёрного ящика»*	Нейросетевая модель распознавания изображений	ART, BasicIterativeMethod
MIM – Momentum Iterative Method	2018	Атака «белого ящика» или «чёрного ящика»*	Нейросетевая модель распознавания изображений	CleverHans library (https://github.com/cleverhans-lab/cleverhans), momentum_iterative_method
The Cube Attack	2019	Атака «белого ящика»	Ансамбль решающих деревьев, задача распознавания изображений	https://github.com/max-andr/provably-robust-boosting
RBA – Region-Based Attack	2019	Атака «белого ящика»	Ансамбль решающих деревьев, различные задачи (austr, cancer, covtype, diabetes, mnist, fourclass, halfmoon)	https://github.com/chenhongge/RobustTrees
HSJA – Hop Skip Jump Attack	2019	Атака «чёрного ящика»	Нейросетевая модель и модель типа «случайный лес», задача распознавания изображений	ART, HopSkipJump
Sign-OPT	2020	Атака «чёрного ящика»	Нейросетевая модель распознавания изображений	ART, SignOPTAttack
LT – Leaf Tuple	2020	Атака «белого ящика»	Ансамбль решающих деревьев, различные задачи (breast-cancer, diabetes, MNIST, ijcnn, webspam, covtype, bosch, HIGGS)	https://github.com/chong-z/tree-ensemble-attack

Figure 35 - Adversarial attacks and example implementations [33].

Every evasion attack uses special methods to create adversarial examples, which can be seen as masks or data changes (e.g., in the case of the above example with the panda image - how to find the hidden mask in the image?). Each of the attacks presented in the table has its own unique way of creating adversarial examples.

White-box attacks use knowledge of the internals of the target model to create adversarial examples that fool the model. Neural networks and decision trees use different methods to find such examples. Whereas black box attacks have no information about the internals of the target model. Most studies focus on white-box attacks, but some authors also investigate the transition from white-box to black-box attacks using the adversarial example portability property. This property allows adversarial examples to retain their effectiveness when used against other models. Implementing a black-box attack involves training a “replacement model,” creating adversarial examples for that model, and applying them to the original model. Although rigorous evidence and explanations for portability have not yet been established, numerous studies have confirmed this property on a variety of datasets.

Since the random forest model is widely used as a classifier, it is important to investigate its robustness to adversarial attacks. However, it is known that classical black-box attacks do not take into account the specificity of solver trees. In the case of ensembles of decision trees, it is impossible to apply typical white-box attacks that are successfully used against neural networks. This is because the loss function in a random forest model is usually a discontinuous piecewise defined function for which no gradient exists, making it difficult or impossible to apply gradient-based attacks to such models.

Because of these factors, the design of intrusion detection systems that utilize machine learning techniques must pay special attention to the study of attacks that target specific models, such as ensembles of decision trees (in particular, the “random forest”).

Since adversarial distortion attacks rely on the use of gradient lifting to find instances of the adversarial space, the general idea of defending machine learning models against such attacks is to make it more difficult for the adversary to gain information about the gradients of the model's decision surface.

Traditional methods for improving the robustness of machine learning models, such as weight reduction, generally do not provide practical protection against malicious examples. To date, only two methods have shown some significant protection.

Adversarial training is one possible method of defense against distortion attacks. If a machine learning model is trained on malicious examples, it can minimize the adversarial space available to attackers. This defense method tries to cover all possible input variants for the classifier, while using data samples that belong to a theoretical input space that is not covered by the original training data distribution. Models trained in this way should ideally not be fooled by malicious examples known to them, but can this method allow defeating an attacker at his own game? - is an open question.

Adversarial learning has shown good results in studies by experts, but it does not solve the problem completely, as the success of this defense method depends on a constant race between the attacking and defending parties.

Therefore, it is infeasible to cover all possible input variants, and an experienced attacker with sufficient computational resources will most likely always be able to find malicious examples that were not used to train the model.

Another technique for defending against distortion attacks is defensive distillation.

Distillation was originally developed to reduce the size of neural network models and reduce the computational resource requirements so that they can run on resource-constrained devices such as mobile devices. This was achieved by training an optimized model using replacement of categorical class labels from the original dataset with probabilistic outcome vectors on a simpler model. The resulting model had a smoother decision surface, which in turn made it more difficult for attackers to obtain the desired gradient.

However, like adversarial learning, distillation only slows down and complicates the process of detecting and exploiting adversarial spaces, so it only provides some protection against attackers with limited computational resources.

Defensive distillation is a technique originally developed to make neural networks more resilient to attacks using machine learning techniques. However, in the context of XGBoost or other tree-busting algorithms, this technique is not directly applicable due to differences in the architecture and operating principles of these models.

Applying this technique to algorithms such as random forests requires some adaptation, as the architecture and operating principles of these models are significantly different. In addition, this method can be computationally expensive.

Using defensive distillation for random forests is an experimental technique and requires careful testing and tuning. Alternative methods for improving robustness, such as regularization and the use of ensembles, may also be useful and easier to implement.

It is difficult to defend against distortion attacks because of the problem of imperfect learning, where statistical processes cannot capture all possible inputs needed for correct classification. In most cases, machine learning models perform very well, but only work with a small number of all possible inputs they may encounter.

Developing a strategy capable of providing a defense against a powerful and adaptive attacker is an important area of research for machine learning practitioners.

Adversarial examples show that many state-of-the-art machine learning algorithms can be hacked in unconventional ways. These machine learning failures demonstrate that even simple algorithms can behave quite differently than their designers intended.

3.2 Application of adversarial learning in intrusion detection algorithms to realize protection against attacks

This subsection implements adversarial learning defense against evasion attacks targeting ML-based IDSs.

The implementation consists of two main steps:

1. Evasion attack execution: creating adversarial patterns for the model;
2. Adversarial learning: extending the original dataset with correctly labeled adversarial samples and training a new and adversarial resistant model on the new training set.

A brief description of the implementation of the main steps:

1. Searching for adversarial samples:
   - Each model is trained on the CICIDS2017 dataset (in the web attacks subset: web_attacks_balanced.csv).
   - The performance of the models is evaluated on the test set.
   - For all samples that are correctly labeled as an attack by the model, the value of the “Total Forward Packet Length” function changes within the specified range.

– If the model changes its prediction for a sample with a changed “Total Fwd Packet Length” function, that sample is adversarial (i.e., it misleads the model).

– A second test set with adversarial samples is generated. The performance of the model is evaluated on this test set. The performance is expected to decrease: even one adversarial sample provides an opportunity for an attack.

2. Defense against evasion attack:

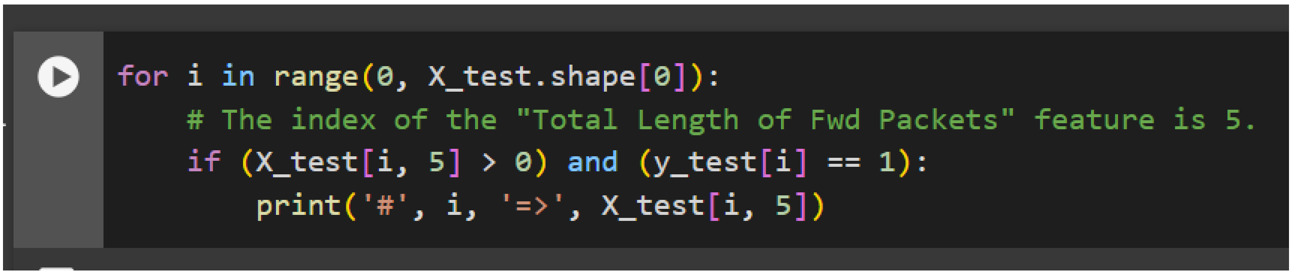
– Adversarial samples are labeled as “attacking” and added to the original training and test sets.

– A new model is trained on a new training set.

– The performance of the adversarially trained model is evaluated on the new test set. The performance is expected to be close to that of the original model before the attack, since the addition of adversarial samples increases the robustness of the model to adversarial attacks.

We will show some aspects of the practical implementation of the evasion attack on the previously proposed models by the authors. Let us consider VoitingModel as an example.

In order to implement the evasion attack, we first need to modify the feature “Total Length of Fwd Packets” (index 5 in the list of selected features) with the check of non-zero values of the feature for samples with the type “attack” (Figure 36).



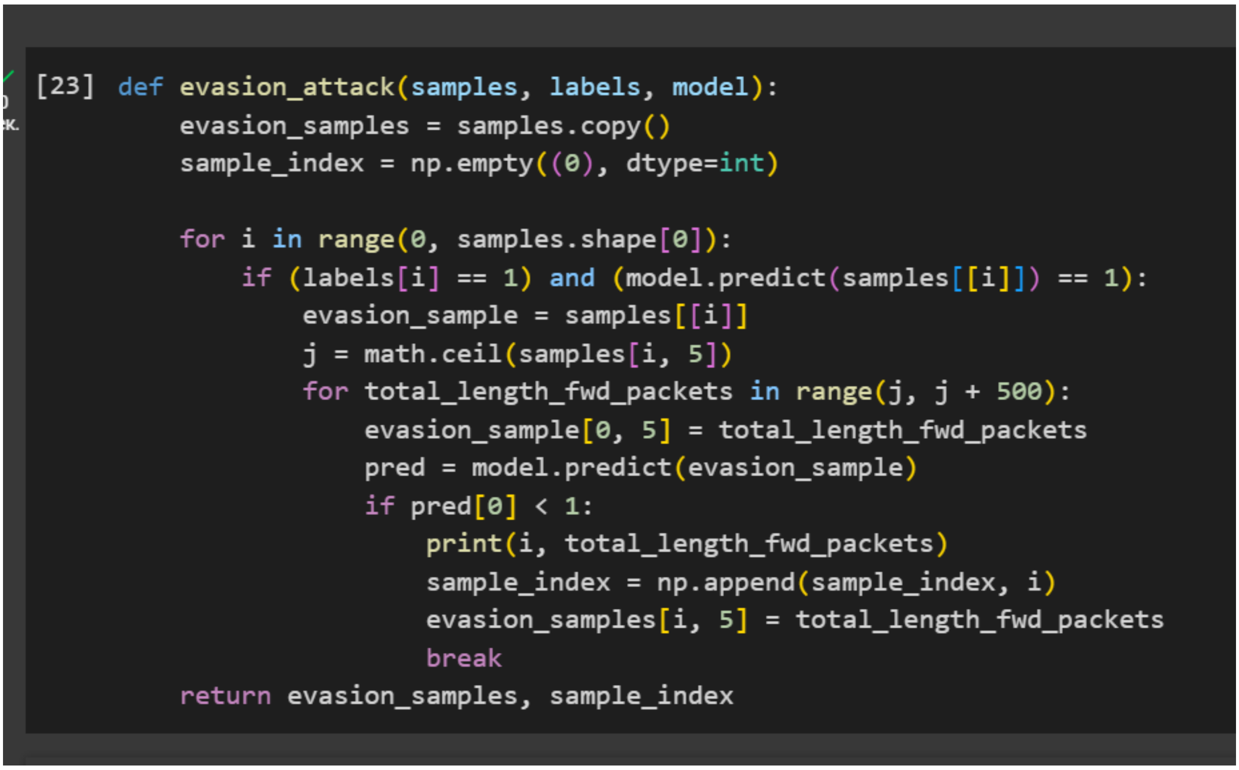
```
for i in range(0, X_test.shape[0]):
    # The index of the "Total Length of Fwd Packets" feature is 5.
    if (X_test[i, 5] > 0) and (y_test[i] == 1):
        print('#', i, '=>', X_test[i, 5])
```

Figure 36- Checking for non-zero values of a trait.

The evasion_attack function finds adversarial samples for the given samples. It returns a copy of the given feature matrix with the found adversarial samples that replaced the original ones in the matrix and the indices of these samples (Figure 37).

The function works as follows:

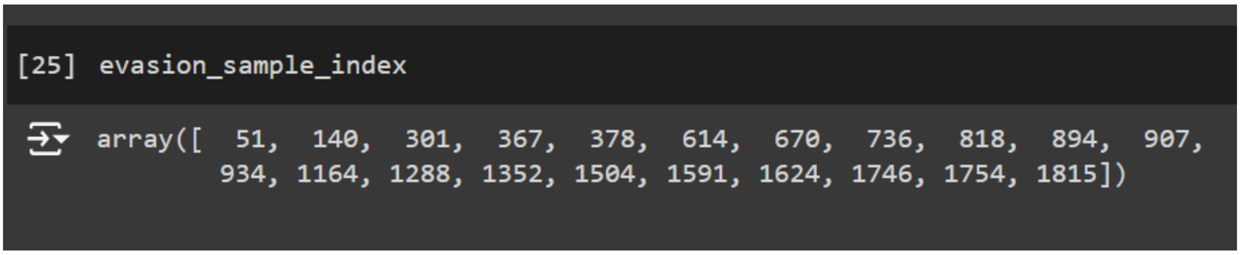
- all samples that are correctly labeled by the model as an attack are processed;
- for these samples, the value of the “Total Length of Fwd Packets” feature is changed in the range [initial value, initial value + 500);
- if the model changes its prediction for a sample with the “Total Length of Fwd Packets” trait changed, that sample is adversarial. The function outputs the index and the new value of “Total Packet Length Fwd” of that sample.



```
[23] def evasion_attack(samples, labels, model):
         evasion_samples = samples.copy()
         sample_index = np.empty((0), dtype=int)

         for i in range(0, samples.shape[0]):
             if (labels[i] == 1) and (model.predict(samples[[i]]) == 1):
                 evasion_sample = samples[[i]]
                 j = math.ceil(samples[i, 5])
                 for total_length_fwd_packets in range(j, j + 500):
                     evasion_sample[0, 5] = total_length_fwd_packets
                     pred = model.predict(evasion_sample)
                     if pred[0] < 1:
                         print(i, total_length_fwd_packets)
                         sample_index = np.append(sample_index, i)
                         evasion_samples[i, 5] = total_length_fwd_packets
                         break
         return evasion_samples, sample_index
```

Figure 37 - Evasion_attack function.

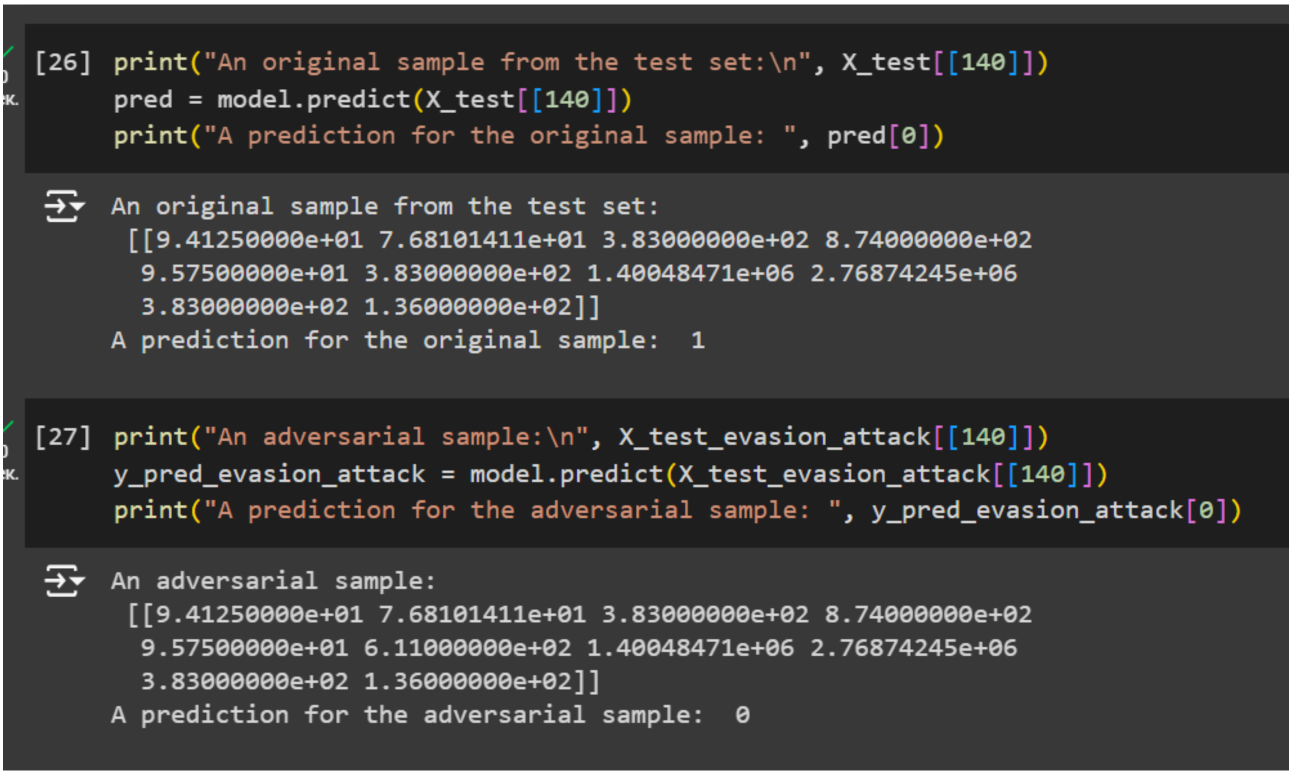
It was found that the found adversarial samples for the original samples are under the following indices (Figure 38).



```
[25] evasion_sample_index

array([  51,  140,  301,  367,  378,  614,  670,  736,  818,  894,  907,
        934, 1164, 1288, 1352, 1504, 1591, 1624, 1746, 1754, 1815])
```

Figure 38 - Adversarial sample indices.

The difference of predictions with the example of the original sample and its adversarial replacement is presented in Figure 39.



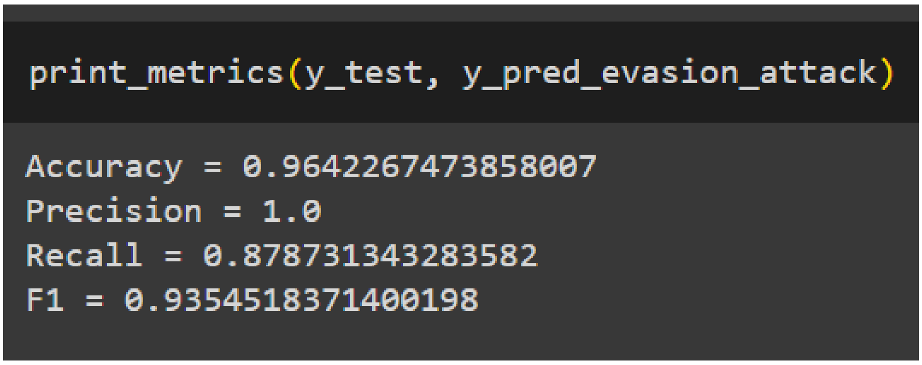
```
[26] print("An original sample from the test set:\n", X_test[[140]])
     pred = model.predict(X_test[[140]])
     print("A prediction for the original sample: ", pred[0])

     An original sample from the test set:
      [[9.41250000e+01 7.68101411e+01 3.83000000e+02 8.74000000e+02
        9.57500000e+01 3.83000000e+02 1.40048471e+06 2.76874245e+06
        3.83000000e+02 1.36000000e+02]]
     A prediction for the original sample:  1

[27] print("An adversarial sample:\n", X_test_evasion_attack[[140]])
     y_pred_evasion_attack = model.predict(X_test_evasion_attack[[140]])
     print("A prediction for the adversarial sample: ", y_pred_evasion_attack[0])

     An adversarial sample:
      [[9.41250000e+01 7.68101411e+01 3.83000000e+02 8.74000000e+02
        9.57500000e+01 6.11000000e+02 1.40048471e+06 2.76874245e+06
        3.83000000e+02 1.36000000e+02]]
     A prediction for the adversarial sample:  0
```

Figure 39 - Difference between an example of the original design and its adversarial replacement.

Thus, the adversarial sample misleads the model, i.e., the classifier changes its response (for the sample with index 140) from “1” (there is an attack) to “0” (no attack).

Note that the retrieved pattern retains its attackability and is, in fact, an effective adversarial pattern: we can increase the value of the “Total Packet Length Fwd” feature by augmenting the payload with zeros/spaces/etc.

Let us specify the model evaluation metrics for the test data with adversarial samples added.



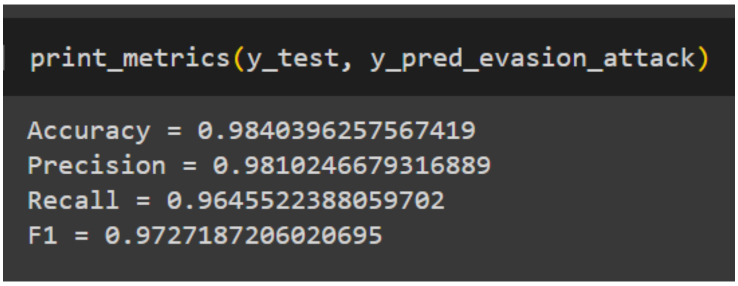
```
print_metrics(y_test, y_pred_evasion_attack)
```

```
Accuracy = 0.9642267473858007
Precision = 1.0
Recall = 0.878731343283582
F1 = 0.9354518371400198
```

Figure 40- VoitingModel evaluation metrics after the implementation of an evasion attack.

For example, some performance metrics deteriorate after the attack because adversarial samples added to the test set mislead the model.

An evasion attack was also implemented on the RF + neural network algorithm. The evaluation results of this model for the test data with adversarial samples added are shown in Figure 41.

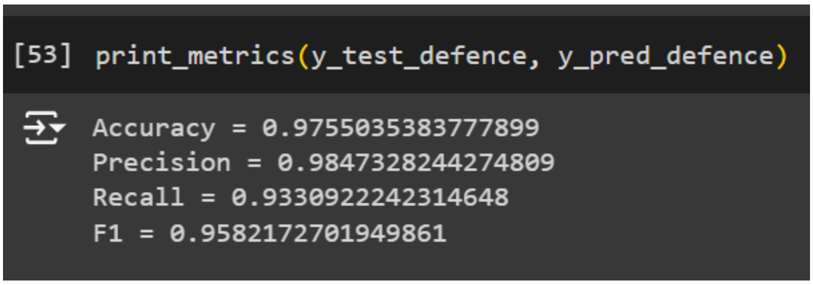


```
print_metrics(y_test, y_pred_evasion_attack)
```

```
Accuracy = 0.9840396257567419
Precision = 0.9810246679316889
Recall = 0.9645522388059702
F1 = 0.9727187206020695
```

Figure 41 - RF + NN evaluation metrics after implementing the evasion attack.

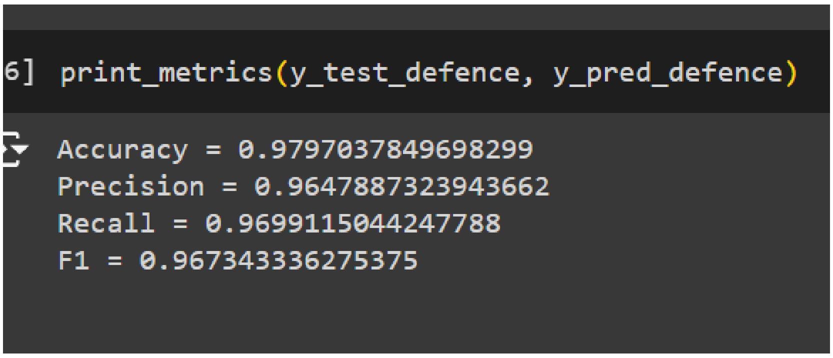
To protect the model from a realized evasion attack, we need to find adversarial samples for the entire dataset and perform adversarial training with them, augmenting the original dataset with adversarial samples that are correctly labeled as “attack”. The post-defense VotingModel (XGB + RF) evaluation metrics are summarized in Figure 42.



```
[53] print_metrics(y_test_defence, y_pred_defence)

     Accuracy = 0.9755035383777899
     Precision = 0.9847328244274809
     Recall = 0.9330922242314648
     F1 = 0.9582172701949861
```

Figure 42 - VotingModel (XGB + RF) evaluation metrics after defense by adversarial learning.

The same process of implementing defense with adversarial learning was performed with the second RF model. The post-protection RF model evaluation metrics are shown in Figure 43.



```
6] print_metrics(y_test_defence, y_pred_defence)

   Accuracy = 0.9797037849698299
   Precision = 0.9647887323943662
   Recall = 0.9699115044247788
   F1 = 0.967343336275375
```

Figure 43 - Metrics for evaluating the RF model after defense by adversarial learning.

To draw conclusions, we summarize the results in a summary table (Table 12).

Table 12 - Performance metrics of the proposed models.

Model name	Model evaluation metrics		
	Before attack	After attack	After defense

VotingModel (XGB + RF)	Accuracy = 0.976
	Precision = 1.0
	Recall = 0.918
	F1 = 0.957
		Accuracy = 0.961
		Precision = 1.0
		Recall = 0.870
		F1 = 0.931
			Accuracy = 0.976
			Precision = 0.985
			Recall = 0.933
			F1 = 0.958

RF + NN	Accuracy = 0.988
	Precision = 0.982
	Recall = 0.978
	F1 = 0.980
		Accuracy = 0.984
		Precision = 0.981
		Recall = 0.965
		F1 = 0.973
			Accuracy = 0.980
			Precision = 0.965
			Recall = 0.970
			F1 = 0.967

Thus, the evaluation metrics of the models are almost restored to the values that were before the evasion attack.

Hence, it can be concluded that the implemented adversarial learning defense improves the robustness of the proposed models against adversarial attacks.

Next, an experiment was conducted by implementing iterative adversarial learning with HopSkipJump attack, which is a powerful black-box evasion attack, and two models, VotingClassifier (Random Forest + XGBClassifier) and RF model with feature extraction with NN using CICIDS2017 dataset.

The hypothesis here is that adversarial learning will then be able to improve the robustness to repeated adversarial attacks. This hypothesis is further disproved by the results.

Let us re-emphasize the main aspects of the raw data used. The CICIDS2017 dataset is prepared by the Canadian Cybersecurity Institute by analyzing network traffic in an isolated environment in which the actions of 25 legitimate users as well as malicious actions of intruders were simulated.

Each record in the CICIDS2017 dataset represents a network session and is characterized by 84 attributes, such as source and destination IP addresses of the data stream (“Source IP” and “Destination IP”), data flow rate (“Flow Bytes/s”), and so on.

Among the 14 types of attacks presented in the dataset, in this study, we consider only web-based attacks. The training subsample contains 4 classes: “BENIGN” (background traffic without attacks, 5087 records), “Web Attack - Brute Force” (1507 records), “Web Attack - Sql Injection” (21 records), “Web Attack - XSS” (652 records).

Note that the task of detecting network attacks on the CICIDS2017 dataset is currently only addressed in 5 papers. In total, the task of “Network Intrusion Detection” on different datasets combines 34 articles, and in total there are more than 100000 articles published on paperswithcode.com.

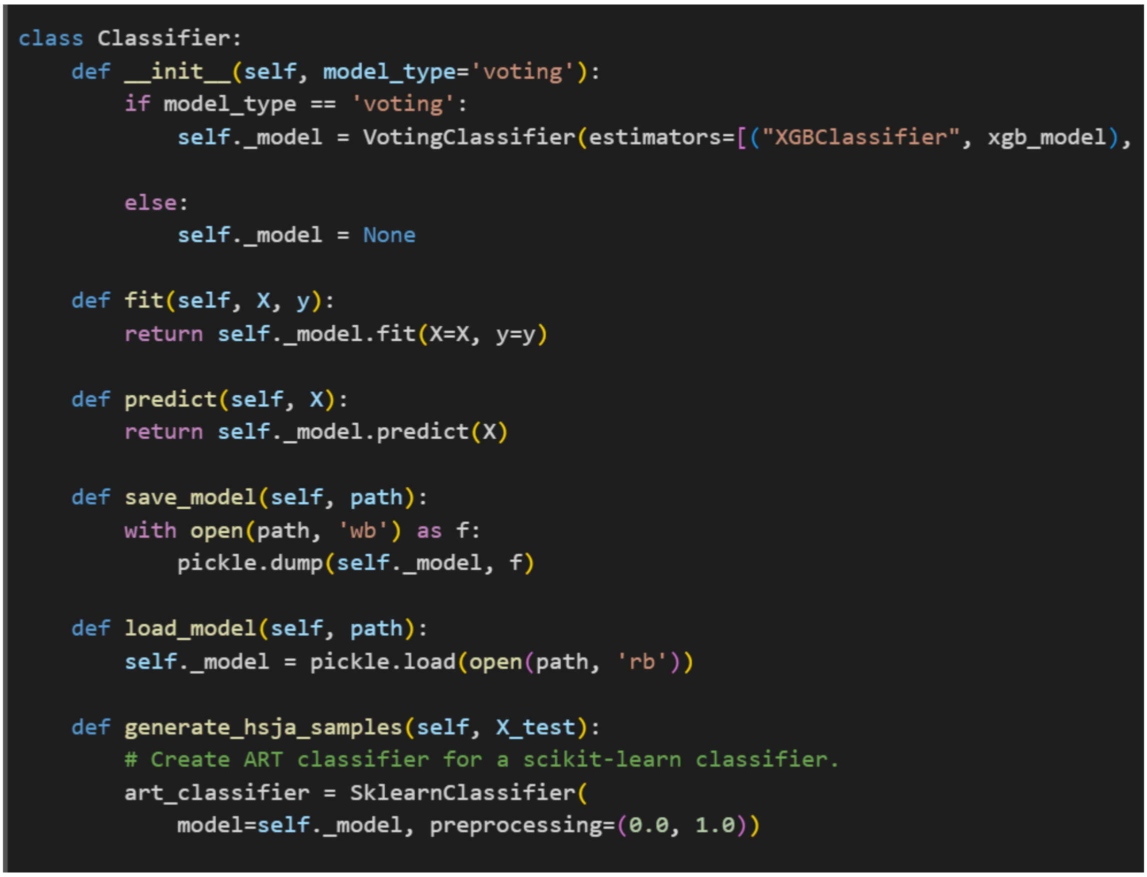
We implement iterative adversarial learning using HopSkipJump attack, which is a powerful black-box evasion attack, and the above models.

The implementation of the experiment is 10 iterations of the Hop Skip Jump attack followed by defense by adversarial learning. This experiment verifies the conclusions of one of the papers (“The Limitations of Deep Learning in Adversarial Settings”), in which it was demonstrated that the model's resistance to adversarial attacks increased after adversarial learning: the repeated attack of the neural network using the JSMA algorithm lost effectiveness; in particular, the number of adversarial examples found decreased from 18000 to 9000.

We use an implementation of the adversarial evasion attack HSJA from the ART library: art.attacks.evasion.HopSkipJump.

Let's first look at the VotingClassifier (Random Forest + XGBClassifier) model. Let's highlight some implementation details.

For convenience, we first defined a classifier class (Figure 44), which will allow us to use the same training code with different classifiers if necessary in the future. It provides the necessary functionality: creating, fitting, saving, loading a model; predicting labels; creating adversarial samples for the original samples.



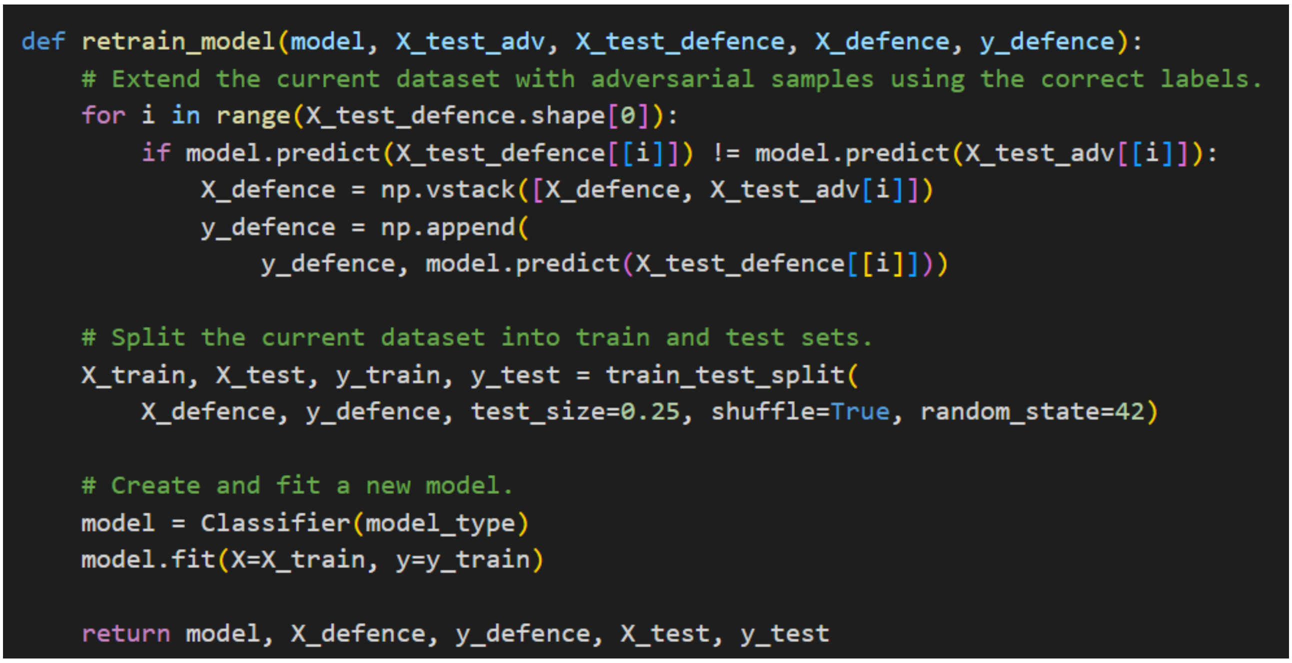
```
class Classifier:
    def __init__(self, model_type='voting'):
        if model_type == 'voting':
            self._model = VotingClassifier(estimators=[("XGBClassifier", xgb_model),

        else:
            self._model = None

    def fit(self, X, y):
        return self._model.fit(X=X, y=y)

    def predict(self, X):
        return self._model.predict(X)

    def save_model(self, path):
        with open(path, 'wb') as f:
            pickle.dump(self._model, f)

    def load_model(self, path):
        self._model = pickle.load(open(path, 'rb'))

    def generate_hsja_samples(self, X_test):
        # Create ART classifier for a scikit-learn classifier.
        art_classifier = SklearnClassifier(
            model=self._model, preprocessing=(0.0, 1.0))
```

Figure 44 - class Classifier.

The key function is generate_hsja_samples - creates adversarial samples for given initial samples using the HopSkipJump attack. It uses the HSJA l2 version implemented in the Adversarial Robustness Toolbox library [34].

The evaluation metrics are accuracy, precision, recall, f1_score.

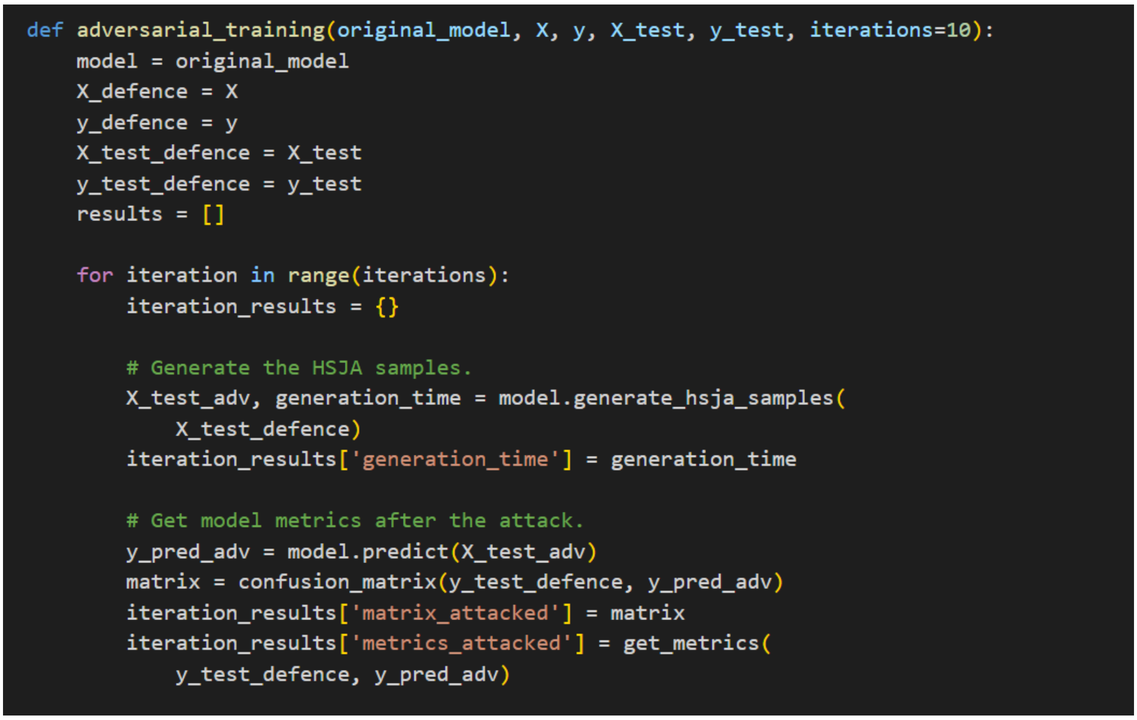
A function is defined that retrains the model using the adversarial samples generated (Figure 45). This step is performed at each iteration of adversarial learning.



```python
def retrain_model(model, X_test_adv, X_test_defence, X_defence, y_defence):
    # Extend the current dataset with adversarial samples using the correct labels.
    for i in range(X_test_defence.shape[0]):
        if model.predict(X_test_defence[[i]]) != model.predict(X_test_adv[[i]]):
            X_defence = np.vstack([X_defence, X_test_adv[i]])
            y_defence = np.append(
                y_defence, model.predict(X_test_defence[[i]]))

    # Split the current dataset into train and test sets.
    X_train, X_test, y_train, y_test = train_test_split(
        X_defence, y_defence, test_size=0.25, shuffle=True, random_state=42)

    # Create and fit a new model.
    model = Classifier(model_type)
    model.fit(X=X_train, y=y_train)

    return model, X_defence, y_defence, X_test, y_test
```

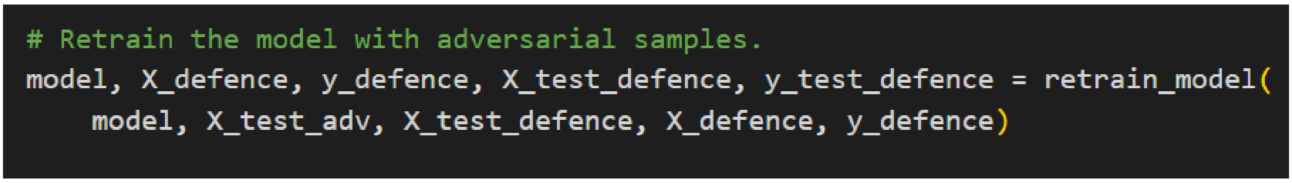
Figure 45 - Retrain_model function.

Next, an iterative adversarial learning algorithm (function adversarial_training) is defined that performs adversarial training of the given model for a given number of iterations using HSJA (Figures 46, 47).



```python
def adversarial_training(original_model, X, y, X_test, y_test, iterations=10):
    model = original_model
    X_defence = X
    y_defence = y
    X_test_defence = X_test
    y_test_defence = y_test
    results = []

    for iteration in range(iterations):
        iteration_results = {}

        # Generate the HSJA samples.
        X_test_adv, generation_time = model.generate_hsja_samples(
            X_test_defence)
        iteration_results['generation_time'] = generation_time

        # Get model metrics after the attack.
        y_pred_adv = model.predict(X_test_adv)
        matrix = confusion_matrix(y_test_defence, y_pred_adv)
        iteration_results['matrix_attacked'] = matrix
        iteration_results['metrics_attacked'] = get_metrics(
            y_test_defence, y_pred_adv)
```

Figure 46 - Part of the adversarial_training function.

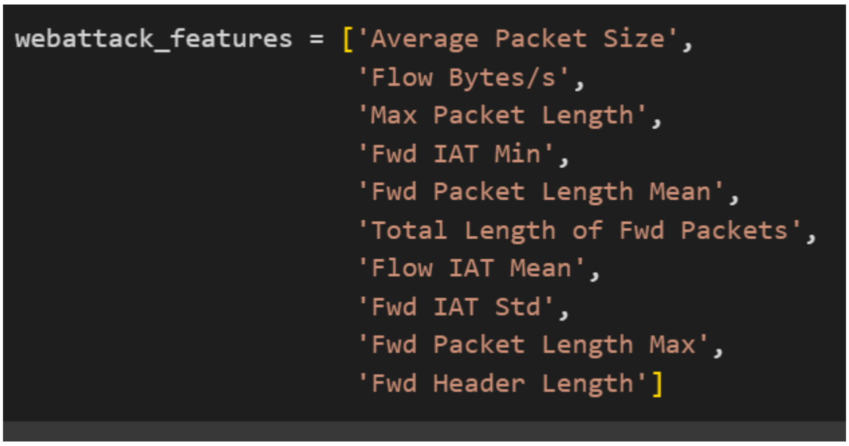


```
# Retrain the model with adversarial samples.
model, X_defence, y_defence, X_test_defence, y_test_defence = retrain_model(
    model, X_test_adv, X_test_defence, X_defence, y_defence)
```

Figure 47 - Part of the adversarial_training function.

Based on the results of the correlation analysis in Chapter 2 of this study, the following features were excluded from the feature space: “Packet Length Mean”, “Subflow Fwd Bytes”, “Avg Fwd Segment Size”, “Fwd IAT Total”, “Fwd Packets/s”, “Fwd IAT Max”. After excluding the features with the lowest significance, the feature space was reduced to a union of 10 features (Figure 48):

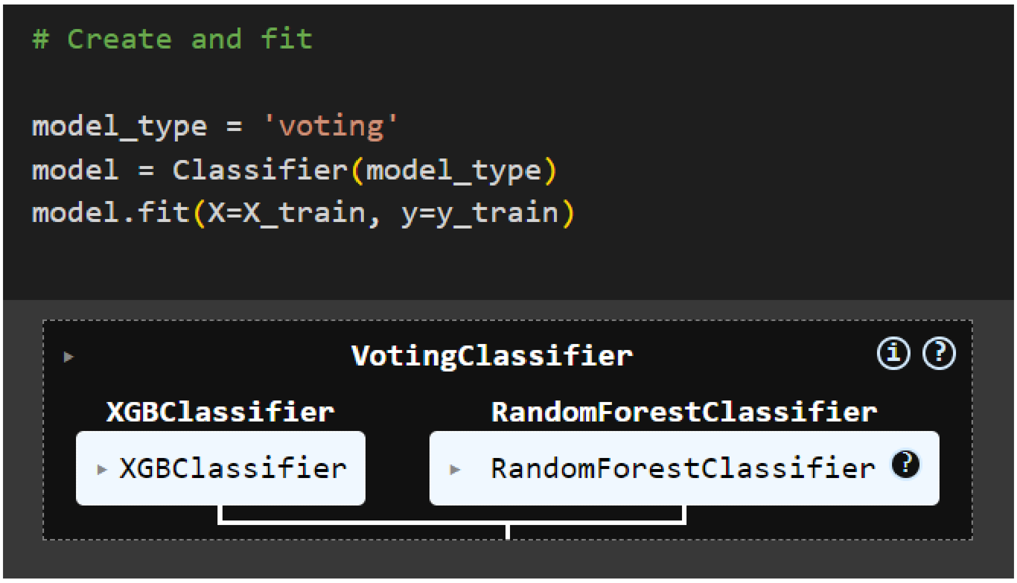
1. “Average Packet Size”, the average length of the data field of a TCP/IP packet (hereafter referred to as packet length).
2. “Flow Bytes/s”, the data flow rate.
3. “Max Packet Length”, the maximum packet length.
4. “Fwd Packet Length Mean”, the average length of packets transmitted in the forward direction.
5. “Fwd IAT Min”, the minimum forward inter-packet interval time (IAT, inter-arrival time) value.
6. “Total Length of Fwd Packets”, the total length of packets transmitted in the forward direction.
7. “Fwd IAT Std”, the standard deviation of the inter-packet interval value in the forward direction of packets.
8. “Flow IAT Mean”, the average value of the inter-packet interval.
9. “Fwd Packet Length Max”, the maximum length of the packet transmitted in the forward direction.
10.“Fwd Header Length”, the total header length of packets transmitted in the forward direction.



```
webattack_features = ['Average Packet Size',
                      'Flow Bytes/s',
                      'Max Packet Length',
                      'Fwd IAT Min',
                      'Fwd Packet Length Mean',
                      'Total Length of Fwd Packets',
                      'Flow IAT Mean',
                      'Fwd IAT Std',
                      'Fwd Packet Length Max',
                      'Fwd Header Length']
```

Figure 48 - 10 most important features.

Training of the model using the previously allocated Classifier class (Figure 49).

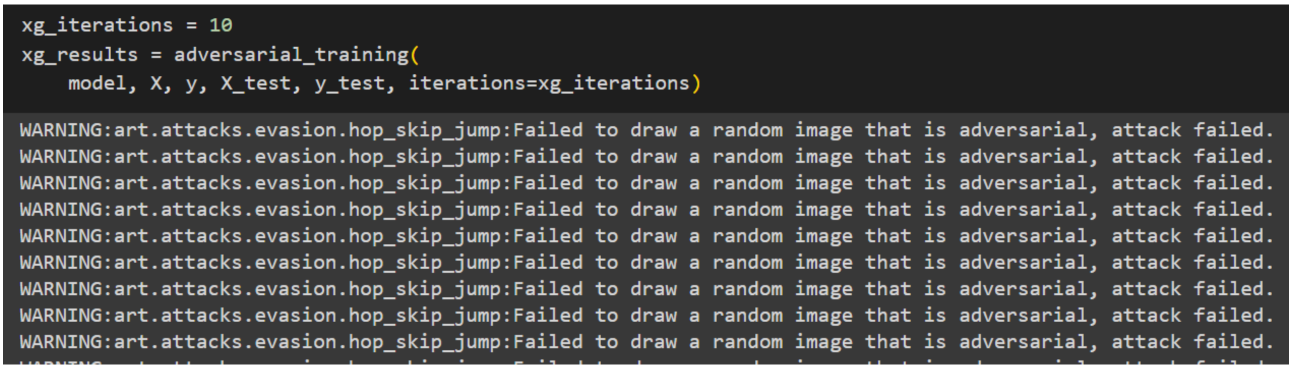


```
# Create and fit

model_type = 'voting'
model = Classifier(model_type)
model.fit(X=X_train, y=y_train)
```

```
VotingClassifier
XGBClassifier          RandomForestClassifier
XGBClassifier          RandomForestClassifier
```

Figure 49 - VotingClassifier training.

A total of 10 iterations were performed. At each iteration, adversarial examples are searched for the available test sample using HSJA (Figure 50). Adversarial learning on the original model for a given number of iterations took a significant amount of time due to the performance of the available hardware. The time cost may vary depending on the power of computing resources.

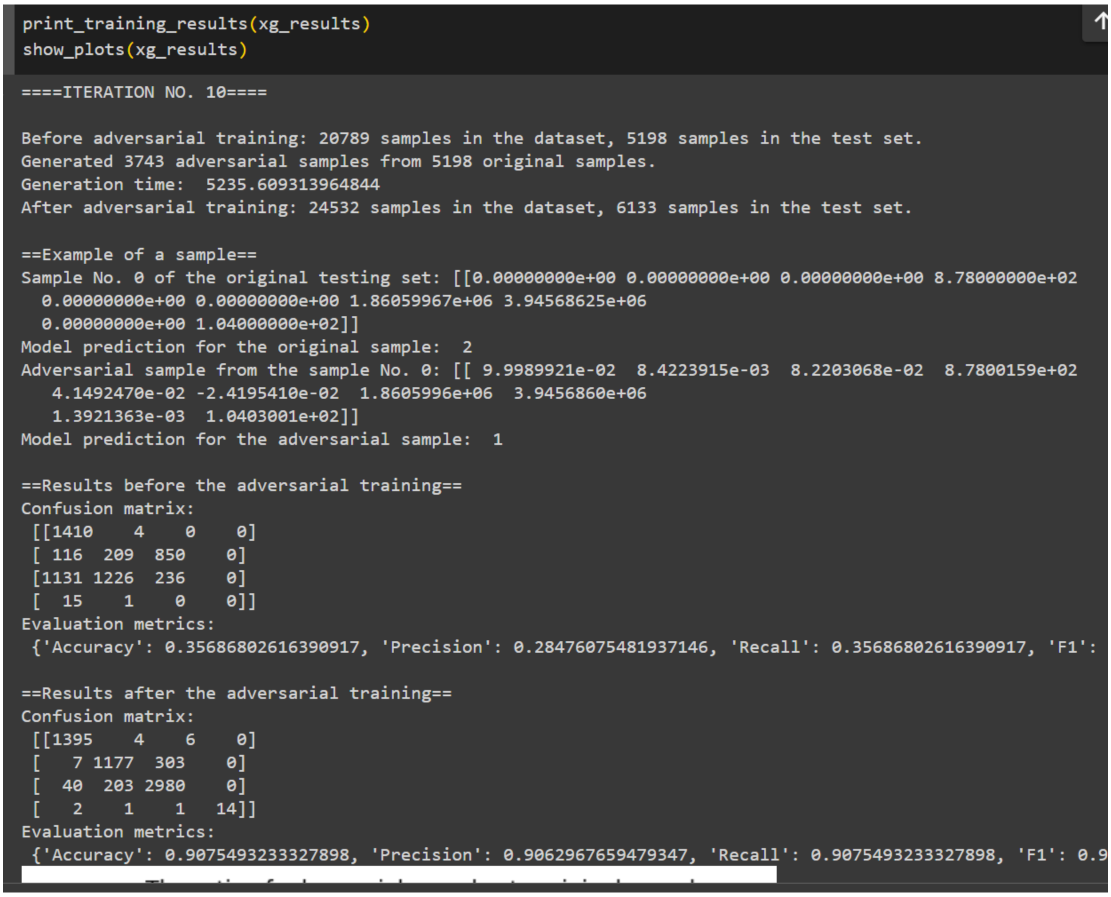


```
xg_iterations = 10
xg_results = adversarial_training(
    model, X, y, X_test, y_test, iterations=xg_iterations)
```

```
WARNING:art.attacks.evasion.hop_skip_jump:Failed to draw a random image that is adversarial, attack failed.
WARNING:art.attacks.evasion.hop_skip_jump:Failed to draw a random image that is adversarial, attack failed.
WARNING:art.attacks.evasion.hop_skip_jump:Failed to draw a random image that is adversarial, attack failed.
WARNING:art.attacks.evasion.hop_skip_jump:Failed to draw a random image that is adversarial, attack failed.
WARNING:art.attacks.evasion.hop_skip_jump:Failed to draw a random image that is adversarial, attack failed.
WARNING:art.attacks.evasion.hop_skip_jump:Failed to draw a random image that is adversarial, attack failed.
WARNING:art.attacks.evasion.hop_skip_jump:Failed to draw a random image that is adversarial, attack failed.
WARNING:art.attacks.evasion.hop_skip_jump:Failed to draw a random image that is adversarial, attack failed.
WARNING:art.attacks.evasion.hop_skip_jump:Failed to draw a random image that is adversarial, attack failed.
```

Figure 50 - Implementation of iterative adversarial learning.

At each iteration, adversarial examples are searched for in the available test sample using HSJA, after which the found adversarial examples are labeled and added to the original sample. The original sample is divided into training sample (75%) and test sample (25%), after which the model is retrained and the quality is evaluated on the test sample. An example of iteration output is shown in Figure 51.



```
print_training_results(xg_results)
show_plots(xg_results)
```

```
====ITERATION NO. 10====

Before adversarial training: 20789 samples in the dataset, 5198 samples in the test set.
Generated 3743 adversarial samples from 5198 original samples.
Generation time:  5235.609313964844
After adversarial training: 24532 samples in the dataset, 6133 samples in the test set.

==Example of a sample==
Sample No. 0 of the original testing set: [[0.00000000e+00 0.00000000e+00 0.00000000e+00 8.78000000e+02
  0.00000000e+00 0.00000000e+00 1.86059967e+06 3.94568625e+06
  0.00000000e+00 1.04000000e+02]]
Model prediction for the original sample:  2
Adversarial sample from the sample No. 0: [[ 9.9989921e-02  8.4223915e-03  8.2203068e-02  8.7800159e+02
   4.1492470e-02 -2.4195410e-02  1.8605996e+06  3.9456860e+06
   1.3921363e-03  1.0403001e+02]]
Model prediction for the adversarial sample:  1

==Results before the adversarial training==
Confusion matrix:
 [[1410    4    0    0]
 [ 116  209  850    0]
 [1131 1226  236    0]
 [  15    1    0    0]]
Evaluation metrics:
 {'Accuracy': 0.35686802616390917, 'Precision': 0.28476075481937146, 'Recall': 0.35686802616390917, 'F1':

==Results after the adversarial training==
Confusion matrix:
 [[1395    4    6    0]
 [   7 1177  303    0]
 [  40  203 2980    0]
 [   2    1    1   14]]
Evaluation metrics:
 {'Accuracy': 0.9075493233327898, 'Precision': 0.9062967659479347, 'Recall': 0.9075493233327898, 'F1': 0.9
```

Figure 51 - Output of the results for the 10th iteration.

The results for 10 iterations of adversarial learning are presented in Table 13.

Table 13 - Results for 10 iterations of adversarial learning with VotingModel and HSJA attack.

No. of iterations	Generation time, sec	Metrics before protection	Metrics after protection

1	937.8	Accuracy=0.715
		Precision=0.648
		Recall=0.715
		F1=0.630
			Accuracy=0.871
			Precision=0.870
			Recall=0.871
			F1=0.870

2	1044,9	Accuracy=0.761
		Precision=0.747
		Recall=0.761
		F1=0.723
			Accuracy=0.864
			Precision=0.859
			Recall=0.864
			F1=0.860

3	1273.8	Accuracy=0.702
		Precision=0.657
		Recall=0.702
		F1=0.658
			Accuracy=0.863
			Precision=0.863
			Recall=0.863
			F1=0.862

4	2115.5	Accuracy=0.460
		Precision=0.556
		Recall=0.460
		F1=0.480
			Accuracy=0.869
			Precision=0.868
			Recall=0.869
			F1=0.868

5	1881.2	Accuracy=0.591
		Precision=0.571
		Recall=0.591
		F1=0.504
			Accuracy=0.873
			Precision=0.869
			Recall=0.873
			F1=0.870

6	2268.6	Accuracy=0.560
		Precision=0.505
		Recall=0.560
		F1=0.489
			Accuracy=0.875
			Precision=0.873
			Recall=0.875
			F1=0.874

Continuation of Table 13.

No. of iterations	Generation time, sec	Metrics before protection	Metrics after protection

7	2820.1	Accuracy=0.478
		Precision=0.437
		Recall=0.478
		F1=0.369
			Accuracy=0.870
			Precision=0.867
			Recall=0.870
			F1=0.869

8	3401.6	Accuracy=0.437
		Precision=0.413
		Recall=0.437
		F1=0.335
			Accuracy=0.882
			Precision=0.880
			Recall=0.882
			F1=0.881

9	4165.0	Accuracy=0.404
		Precision=0.328
		Recall=0.404
		F1=0.336
			Accuracy=0.890
			Precision=0.888
			Recall=0.890
			F1=0.888

10	5235.6	Accuracy=0.357
		Precision=0.285
		Recall=0.357
		F1=0.288
			Accuracy=0.910
			Precision=0.906
			Recall=0.910
			F1=0.907

Using a pre-written function we output the following statistics (Figure 52):

- ratio of adversarial samples generated to original samples per iteration;
- number of adversarial samples generated per iteration;
- average time for the generation algorithm to process one original sample per iteration;
- total generation time per iteration;
- performance metrics per iteration for two cases: after attack and after defense.

Graphs of the obtained results are shown below in Figures 52 - 56.

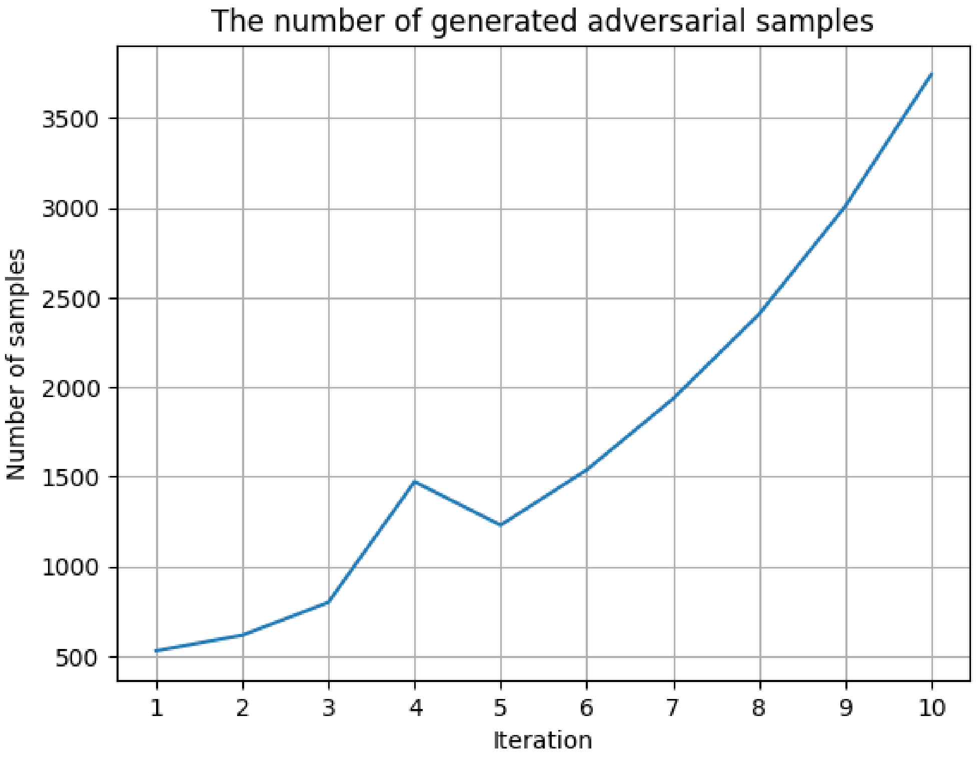


Figure 52- Ratio of adversarial samples generated to original samples per iteration.

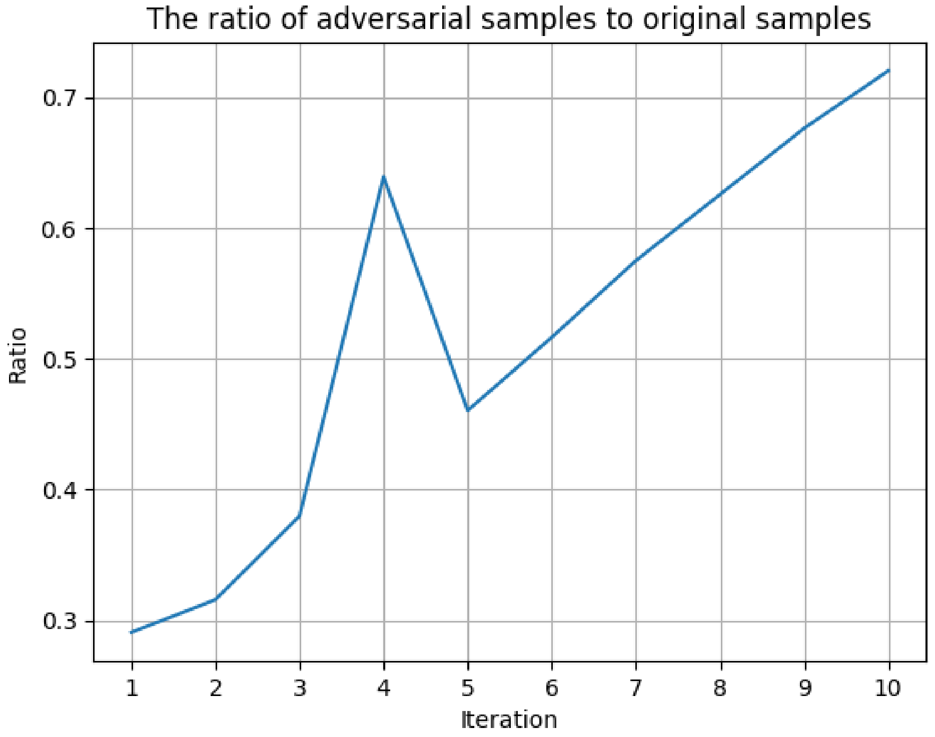


Figure 53 - Number of adversarial samples generated per iteration.

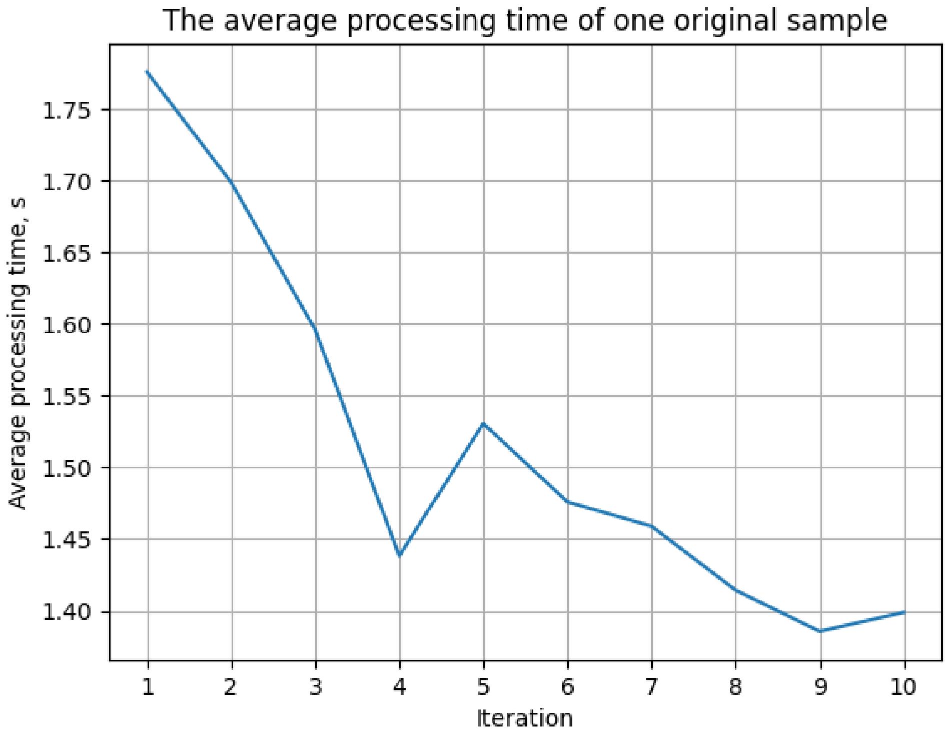


Figure 54 - Average time for the generation algorithm to process one initial sample per iteration.

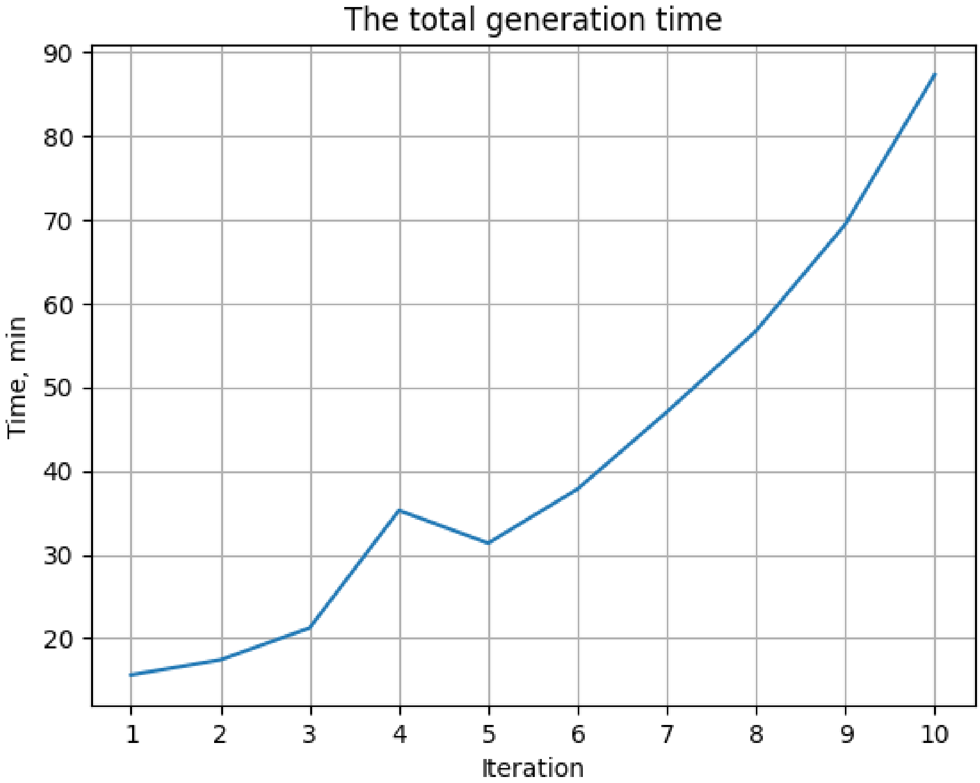


Figure 55 - Total generation time per iteration.

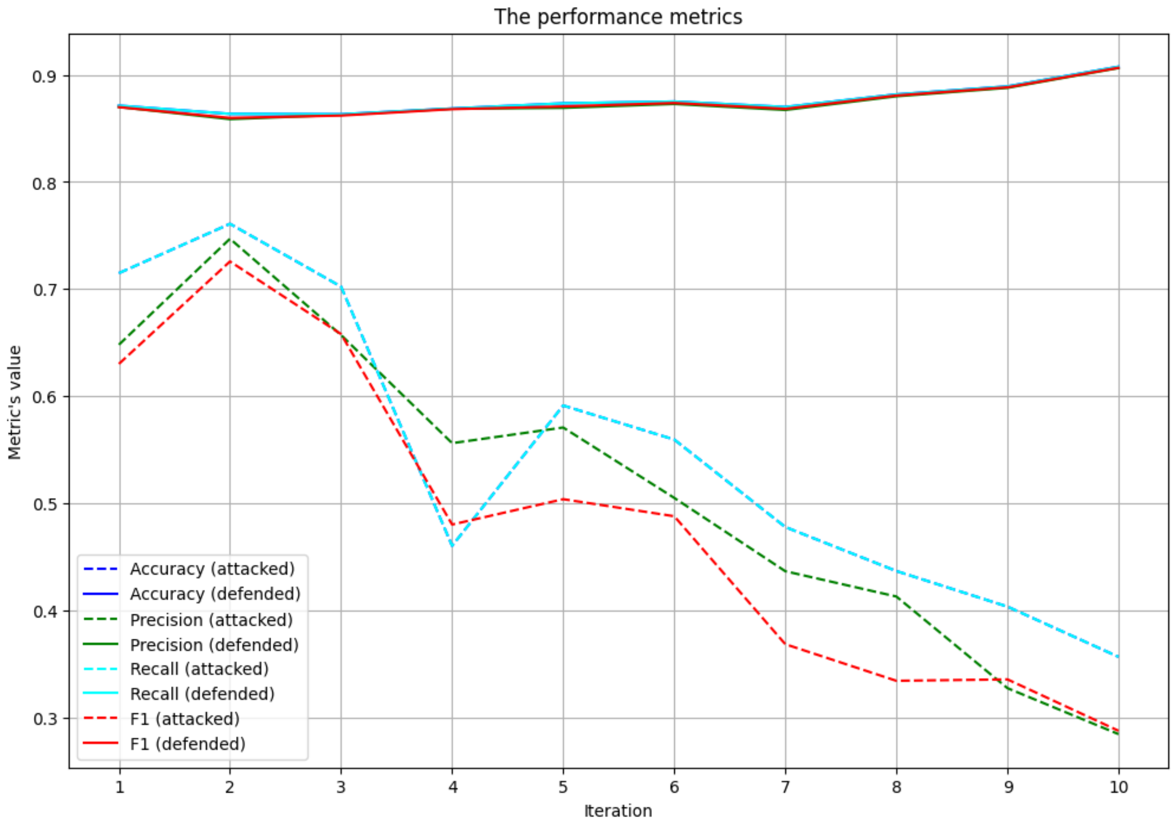


Figure 56 - Performance metrics per iteration for two cases: after attack and after defense.

The results for 10 iterations of adversarial learning for the second model are presented in Table 14.

Table 14 - Results for 10 iterations of adversarial learning with RF and HSJA attack.

No. of iterations	Generation time, sec	Metrics before protection	Metrics after protection

1	920.6	Accuracy=0.767
		Precision=0.649
		Recall=0.718
		F1=0.620
			Accuracy=0.880
			Precision=0.820
			Recall=0.876
			F1=0.867

2	1024.4	Accuracy=0.771
		Precision=0.758
		Recall=0.789
		F1=0.747
			Accuracy=0.869
			Precision=0.860
			Recall=0.865
			F1=0.867

Continuation of Table 14.

No. of iterations	Generation time, sec	Metrics before protection	Metrics after protection

3	1120.8	Accuracy=0.710
		Precision=0.689
		Recall=0.789
		F1=0.710
			Accuracy=0.898
			Precision=0.870
			Recall=0.893
			F1=0.867

4	2328.1	Accuracy=0.510
		Precision=0.569
		Recall=0.501
		F1=0.509
			Accuracy=0.879
			Precision=0.868
			Recall=0.879
			F1=0.868

5	1671.5	Accuracy=0.610
		Precision=0.574
		Recall=0.610
		F1=0.508
			Accuracy=0.886
			Precision=0.870
			Recall=0.886
			F1=0.871

6	2427.7	Accuracy=0.590
		Precision=0.510
		Recall=0.590
		F1=0.497
			Accuracy=0.879
			Precision=0.875
			Recall=0.879
			F1=0.880

7	2910.6	Accuracy=0.490
		Precision=0.443
		Recall=0.481
		F1=0.402
			Accuracy=0.881
			Precision=0.867
			Recall=0.881
			F1=0.869

8	3076.2	Accuracy=0.511
		Precision=0.503
		Recall=0.498
		F1=0.489
			Accuracy=0.892
			Precision=0.880
			Recall=0.892
			F1=0.880

9	4792.4	Accuracy=0.501
		Precision=0.487
		Recall=0.471
		F1=0.407
			Accuracy=0.904
			Precision=0.897
			Recall=0.901
			F1=0.891

10	5932.3	Accuracy=0.453
		Precision=0.397
		Recall=0.401
		F1=0.305
			Accuracy=0.927
			Precision=0.911
			Recall=0.927
			F1=0.910

Results and conclusions of the third chapter

The robustness of the developed models to adversarial attacks was assessed, i.e. how difficult or easy it would be for an attacker to “cheat” the system.

Of the main types of adversarial attacks used was the implementation of an evasion attack, in which an attacker selects input data during the exploitation phase so that the model gives an incorrect response.

Since the random forest model is widely used as a classifier, it is important to investigate its robustness to adversarial attacks. However, it is known that classical black-box attacks do not take into account the specificity of solver trees. In the case of ensembles of decision trees, it is impossible to apply typical white-box attacks that are successfully used against neural networks.

Because of these factors, when developing intrusion detection systems that utilize machine learning techniques, special attention should be paid to studying attacks that target specific models, such as ensembles of decision trees.

Traditional methods for improving the robustness of machine learning models, such as weight reduction, generally do not provide practical protection against malicious examples. To date, only two methods have shown some significant protection - adversarial training and defensive distillation.

Adversarial training has shown good results in studies by experts, but it does not solve the problem completely, because the success of this defense method depends on a constant race between the attacking and defending parties.

It is difficult to defend against distortion attacks because of the problem of imperfect learning, where statistical processes cannot capture all possible inputs needed for correct classification.

Developing a strategy that can defend against a powerful and adaptive attacker is an important research area for machine learning practitioners.

Adversarial examples show that many modern machine learning algorithms can be hacked in unconventional ways. These machine learning failures demonstrate that even simple algorithms can behave quite differently than their designers intended.

A two-step defense against evasion attacks using adversarial learning has been implemented - performing an evasion attack (creating adversarial samples for the model) and adversarial learning (expanding the original dataset with correctly labeled adversarial samples and training a new and adversarial-resistant model on the new training set).

For example, some performance metrics have been shown to deteriorate after the attack is implemented because adversarial samples added to the test set mislead the model. However, after the adversarial training is implemented, they are almost restored to the values they were before the evasion attack.

Hence, it can be concluded that the implemented defense with adversarial learning improves the robustness of the proposed models against adversarial attacks.

An experiment with the implementation of iterative adversarial learning using the HopSkipJump attack and the two models considered: the VotingClassifier (Random Forest + XGBClassifier) and the RF model with feature extraction with NN, disproved the hypothesis that adversarial learning in this case will be able to increase the robustness to repeated adversarial attacks.

Thus, according to the results, the following conclusions can be drawn after conducting this experiment with the models:

1. The performance metrics shift to the worse side after the attack. Adversarial learning after the attack effectively protects the model by recovering the values of the metrics. However, the model remains vulnerable to repeated attacks thereafter: that is, the defense only works against those generated adversarial examples that we correctly labeled and then added to the dataset. A new iteration of the attack can still generate effective adversarial examples.

2. The dataset is expanded at each iteration, which may increase the number of adversarial subsamples generated, but also increases the ratio of adversarial samples to original samples.

3. Adversarial learning did not improve the robustness of the considered models to repeated HSJA attacks.

CONCLUSION

Thus, as a result of this work, the goal was achieved and the objectives were solved.

Two algorithms were proposed, VotingClassifier (Random Forest + XGBClassifier) and RF + neural network for feature extraction. The latter approach showed a higher Accuracy metric (0.988) and the former showed a higher Precision metric (1.0) when trained on the same balanced and preprocessed subsample of WebAttacks web attacks of the CICIDS2017 dataset (70% / 30% ratio of normal to abnormal traffic, 10 most significant features selected after the correlation analysis performed earlier).

The robustness of the developed models to adversarial attacks was assessed, i.e. how difficult or easy it would be for an attacker to “trick” the system

The obtained results indicate the necessity of training the proposed machine learning model on the dataset obtained from the analysis of network traffic in the protected network. Otherwise, when using a pre-trained model, it is mandatory to match the physical structure of the protected network and the network in which the model was trained, as well as the settings of network equipment. At the stage of collecting and preparing the training sample, it is necessary to avoid imbalanced distribution of normal and abnormal records, which may cause overtraining of the model and/or a sharp increase in the number of false positives of the classifier.

It is difficult to defend against distortion attacks due to the problem of imperfect learning, where statistical processes cannot capture all possible inputs needed for correct classification. Developing a strategy that can provide defense against a powerful and adaptive attacker is an important research area for machine learning practitioners.

Adversarial learning has shown good results in the research of experts, but it is worth noting that it does not solve the problem completely, as the success of this defense method depends on a constant race between the attacking and defending parties.

Adversarial examples show that many modern machine learning algorithms can be hacked in unconventional ways. These machine learning failures demonstrate that even simple algorithms can behave quite differently than their designers intended.

Reducing performance requirements is possible through the use of “layered” classifiers that combine fast, low-performance models at the preprocessing stage and efficient, computationally complex models at higher levels.

Implementation of the proposed solutions in real-time (near-real-time) systems implies efficient processing and analysis of high-speed data streams in high-power feature space conditions and is possible only in the presence of a high-performance hardware and software platform.

These circumstances together with the known results of research in the subject area allow us to conclude that it is possible to use machine learning methods to search for anomalies and detect computer attacks.

It should be noted that a promising direction for further research is the development of algorithms for detecting computer attacks based on the use of features independent of the physical structure of the network and the settings of the equipment used, as well as the use of deep learning neural networks (deep learning), which demonstrate better results than other methods in solving a wide range of problems.

In addition, an important aspect of the development of this topic is the study of ways to increase the resistance of machine learning algorithms to attacks, as well as the development of methods for detecting anomalies in real time, taking into account the specifics of modern threats and methods of their covert manifestation.

Thus, the study of intrusion detection system (IDS/IPS) vulnerabilities in algorithms based on statistical and machine learning methods requires a comprehensive approach that includes both theoretical research and experimental studies on real data sets. Only such an approach will make it possible to develop effective and reliable methods for protecting information systems from modern cyber threats. This topic represents an urgent and important challenge in the field of cybersecurity, which requires further research and development.

LIST OF SOURCES USED

1. Глущенко, М. В. IDS / IPS — системы обнаружения и предотвращения вторжений / М. В. Глущенко, А. А. Ширяев, С. А. Глушенко. — Текст: электронный // Концепция «Общества знаний2 в современной науке. — 2019. — С. 115-117. — URL: https://www.elibrary.ru/item.asp?id= 41328677 (дата обращения: 30.04.2023).
2. Ле Куанг Минь, Фан Хью Ань, Нгуен Ань Чуен, Нгуен Чунг Тьен «Интегрированная IDS/IPS модель между открытым источников с улучшением машинного обучения». // https://apni.ru/ URL: https://apni.ru/article/152-integrirovannaya-idsips-model-mezhdu-otkritim (дата обращения: 05.03.2024).
3. Кумага Н.К., Григорьевых А.В. «Проектирование и внедрение системы обнаружения и предотвращения вторжений IDS/IPS в корпоративной сети УГТУ». // URL: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://elib.utmn.ru/jspui/bitstream/ru-tsu/28957/1/miim_2023_238_242.pdf (дата обращения: 05.03.2024).
4. А.Д. Алшаиби, М.М Аль-Ани, А.А. Конев «Сравнительный анализ методов машинного обучения, используемых в системах обнаружения вторжений киберфизических систем». // https://cyberleninka.ru/ URL: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-metodov-mashinnogo-obucheniya-v-zadachah-obnaruzheniya-setevyh-anomaliy (дата обращения: 07.03.2024).
5. Баженов И.О. «Методы интеллектуальных технологий в задачах обнаружения атак в компьютерных сетях». // https://cyberleninka.ru/ URL: https://cyberleninka.ru/article/n/issledovanie-primeneniya-neyronnyh-setey-dlya-obnaruzheniya-nizkointensivnyh-ddos-atak-prikladnogo-urovnya (дата обращения: 10.03.2024).
6. Basinya E.A., Lukina M.S. «Разработка модуля системы обнаружения и предотвращения вторжений». // https://cyberleninka.ru/ URL:

https://cyberleninka.ru/article/n/razrabotka-modulya-sistemy-obnaruzheniya-i-predotvrascheniya-vtorzheniy (дата обращения: 10.03.2024).

7. С. М. Трошина, Н. В. Штуллер «Система обнаружения атак». // URL: http://lib.urfu.ru/mns-urfu/author/11203/source/rinc?page=3 (дата обращения: 10.03.2024).

8. Vinayakumar, R., Soman, K.P., Poornachandrany, P. «Applying convolutional neural network for network intrusion detection (Conference Paper)». // https://www.semanticscholar.org/URL:https://www.semanticscholar.org/paper/Applying-convolutional-neural-network-for-network-Vinayakumar-Soman/38b68ee830fa4b85a5411c8b8de36ba18da64d5a (дата обращения: 10.03.2024).

9. Azizjon, M., Jumabek, A., Kim, W., «1D CNN based network intrusion detection with normalization on imbalanced data». // https://www.researchgate.net/ URL: https://www.researchgate.net/publication/339641880_1D_CNN_Based_Network_Intrusion_Detection_with_Normalization_on_Imbalanced_Data (дата обращения: 11.03.2024).

10. S. Mukkamala, G. Janoski, and A. Sung, «Intrusion detection using neural networks and support vector machines». // https://www.semanticscholar.org/ URL: https://www.semanticscholar.org/paper/Intrusion-detection-using-neural-networks-and-Mukkamala-Janoski/159baa5ef8f325ef736ca7be12d27e8ec96d7542 (дата обращения: 11.03.2024).

11. Gharib, A., Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A, «An Evaluation Framework for Intrusion Detection Dataset». // https://www.semanticscholar.org/ URL: https://www.semanticscholar.org/paper/An-Evaluation-Framework-for-Intrusion-Detection-Gharib-Sharafaldin/11a9ae9a37ac6c96a4344b97165970fd2f594deb (дата обращения: 11.03.2024).

12. R. Perdisci, G. Giacinto, and W. Lee, «Using an ensemble of one-class SVMs for network intrusion detection». // https://www.researchgate.net/ URL: https://www.researchgate.net/publication/220765814_Using_an_Ensemble_of_One-Class_SVM_Classifiers_to_Harden_Payload-based_Anomaly_Detection_Systems (дата обращения: 14.03.2024).

13. Ansam Khraisat, Iqbal Gondal, Peter Vamplew & Joarder Kamruzzaman, «Survey of intrusion detection systems: techniques, datasets and challenges». // https://www.researchgate.net/ URL: https://www.researchgate.net/publication/334533397_Survey_of_intrusion_detection_systems_techniques_datasets_and_challenges (дата обращения: 14.03.2024).

14. H. Kim and S. Kim, «Hybrid anomaly detection system for intrusion detection». // https://www.researchgate.net/ URL: https://www.researchgate.net/publication/259138030_A_novel_hybrid_intrusion_detection_method_integrating_anomaly_detection_with_misuse_detection (дата обращения: 14.03.2024).

15. M. Mayuranathan, M. Murugan & V. Dhanakoti «Best features based intrusion detection system by RBM model for detecting DDoS in cloud environment». // https://www.researchgate.net/ URL: https://www.researchgate.net/publication/337842653_Best_features_based_intrusion_detection_system_by_RBM_model_for_detecting_DDoS_in_cloud_environment (дата обращения: 16.03.2024).

16. Alazab, M., Venkatraman, S., & Watters, P. (2016). A survey on machine learning techniques in wireless sensor networks intrusion detection. IEEE Communications Surveys & Tutorials, 18(2), 860-880. // URL: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://web.archive.org/web/20170829035856id_/http://www.eng.usf.edu/~ibutun/Butun191.pdf (дата обращения: 16.03.2024).

17. Описание изобретения к патенту РФ №2019126640, 22.01.2018. Постоянное обучение для обнаружения вторжения // ЛО Пэнчэн, БРИГГС Ривз Хопп, АХМАД Навид.

18. Описание изобретения к патенту РФ №2017125334, 17.07.2017. Система и способ настройки систем безопасности при DDoS-атаке // Халимоненко Александр Александрович (RU), Тихомиров Антон Владимирович (RU), Коноплев Сергей Валерьевич (RU).

19. Описание изобретения к патенту РФ №2017101441, 17.01.2017. Способ защиты веб-приложений при помощи интеллектуального сетевого экрана с использованием автоматического построения моделей приложений // Носеевич Георгий Максимович (RU), Гамаюнов Денис Юрьевич (RU), Шерварлы Валерия Григорьевна (RU), Каюмов Эмиль Марселевич (RU).

20. Описание изобретения к патенту РФ №2005130257/09, 06.11.2003. Системы и способы предотвращения вторжения для сетевых серверов // СЭМПЛ Чар.

21. Описание изобретения к патенту РФ №2016137336, 19.09.2016. Система и способ автогенерации решающих правил для систем обнаружения вторжений с обратной связью // Кислицин Никита Игоревич (RU).

22. Описание изобретения к патенту US10778705B1, 15.09.2020. Метод обнаружения вторжения на основе глубокого обучения, система и компьютерная программа для веб-приложений.

23. Описание изобретения к патенту US20220124111A1, 21.04.2022, Система обнаружения и смягчения последствий кибербезопасности с использованием машинного обучения и расширенной кореляции данных.

24. В.П.Шкодырев, К.И. Ягафаров, В.А. Баштовенко, Е.Э. Ильина, «Обзор методов обнаружения аномалий в потоках данных», Second Conference on Software Engineering and Information Management, 2017.

25. Зубков Евгений Валерьевич, Белов Виктор Матвеевич «Методы интеллектуального анализа данных и обнаружение вторжений» // Вестник СибГУТИ. 2016. №1 (33). URL: https://cyberleninka.ru/article/n/metody-

intellektualnogo-analiza-dannyh-i-obnaruzhenie-vtorzheniy (дата обращения: 26.03.2024).

26. Метод опорных векторов (SVM) // https://neerc.ifmo.ru/ URL: https://neerc.ifmo.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%BE%D0%BF%D0%BE%D1%80%D0%BD%D1%8B%D1%85_%D0%B2%D0%B5%D0%BA%D1%82%D0%BE%D1%80%D0%BE%D0%B2_(SVM) (дата обращения: 29.03.2024).

27. Андреас Мюллер, Сара Гвидо Введение в машинное обучение с помощью PYTHON. - М.,: 2016-2017. - 393 с.

28. Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2016): 785–794.

29. Weilin Xu, Yanjun Qi and David Evans. Automatically Evading Classifiers: A Case Study on PDF, Malware Classifiers. Network and Distributed Systems Symposium 2016, 21–24 February 2016, San Diego, California.

30. Blaine Nelson et al. Exploiting Machine Learning to Subvert Your Spam Filter. Proceedings of the 1st USENIX Workshop on Large-Scale Exploits and Emergent hreats (2008): 1–9.

31. Carbon Black. Beyond the Hype: Security Experts Weigh in on Artificial Intelligence, Machine Learning and Non-Malware Attacks (2017). https://www.carbonblack.com/2017/03/28/beyond-hype-security-experts-weigh-artificial-intelligence-machine-learning-non-malware-attacks/ (дата обращения: 05.04.2024).

32. Чио К., Фримэн Д. Машинное обучение и безопасность / пер. с анг. А. В. Снастина. – М.: ДМК Пресс, 2020. – 388 с.: ил.

33. EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy Google Inc., Mountain View, CA, Published as a conference paper at ICLR 2015.

34. Мы так и не смогли защитить свою модель машинного обучения от состязательных атак // Хабр URL:

https://habr.com/ru/companies/isp_ras/articles/800751/ (дата обращения: 5.05.2024).

35. Module providing evasion attacks under a common interface. // adversarial-robustness-toolbox URL: https://adversarial-robustness-toolbox.readthedocs.io/en/latest/modules/attacks/evasion.html#hopskipjump-attack (дата обращения: 10.05.2024).

36. Rong-En Fan et al. LIBLINEAR: A Library for Large Linear Classification. Journal of Machine Learning Research 9 (2008): 1871–1874.

37. Francis Bach. Stochastic Optimization: Beyond Stochastic Gradients and Convexity. INRIA – Ecole Normale Supérieure, Paris, France. Joint tutorial with Suvrit Sra, MIT – NIPS, 2016.

38. EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES Ian J. Goodfellow, Jonathon Shlens & Christian Szegedy Google Inc., Mountain View, CA, Published as a conference paper at ICLR 2015.

39. Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, Ananthram Swami. Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks.

40. Иман Шарафальдин, Араш Хабиби Лашкари и Али А. Горбани, «На пути к созданию нового набора данных для обнаружения вторжений и характеристике трафика вторжений», 4-я Международная конференция по безопасности и конфиденциальности информационных систем (ICISSP), Португалия, январь 2018 г.

41. Kahraman Kostas. Anomaly Detection in Networks Using Machine Learning. 2018 (error was found in assessing the importance of features) [Kostas2018].

42. Ger, Alex & Goryunov, M. & Matskevich, A. & Rybolovlev, Dmitry & Nikolskaya, Anastasiya. (2024). Adversarial Attacks Against a Machine Learning Based Intrusion Detection System. 10.48612/jisp/eatr-5pxb-akt8.