MINISTRY OF HIGHER EDUCATION, SCIENCE AND INNOVATION OF THE REPUBLIC OF UZBEKISTAN

TASHKENT STATE UNIVERSITY OF ECONOMICS

DIGITAL TRANSFORMATION AND ARTIFICIAL INTELLIGENCE: PROBLEMS, INNOVATIONS AND TRENDS

1st International Scientific - Practical Conference

CONFERENCE PROCEEDINGS

SEPTEMBER 11, TASHKENT 2024

ORGANIZING COMMITTEE

CHAIRMAN AND DEPUTIES

Tulkin Teshabayev

Rector of Tashkent State University of Economics

Gulnora Abduraxmanova

Vice-rector for scientific affairs and innovations

Sultonali Mehmonov

Vice-rector for academic affairs

Komila Karimova

First rector for youth issues and spiritual and educational affairs

Sherzod Sindarov

Vice-rector for international cooperation

Ulug‘bek Xalikov

Vice-rector for international cooperation

Nodir Akbarov

Dean of the Faculty of Digital Economy

Sanjar Mirzaliyev

Head of the Department of Scientific Research and Innovation

MEMBERS OF THE SCIENTIFIC AND TECHNICAL COMMITTEE

Bahodir Muminov

Head of the Department of Artificial Intelligence, Professor

Dilshod Mirzaev

Head of the Department of Information Systems and Technologies

Diyora Khashimova

Faculty of Digital economy, deputy dean

Rashid Nasimov

Associate professor of the Department of Artificial Intelligence

Sharofutdin Xoshimxodjaev

Associate professor of the Department of Artificial Intelligence

Guzal Belalova

Associate professor of the Department of Artificial Intelligence

Dilmurod Mirzaaxmedov

Senior teacher of the Department of Artificial Intelligence

Sanjar Muhammadiev

Senior teacher of the Department of Artificial Intelligence

Muminbek Khayrullayev

Assistant of the Department of Artificial Intelligence

Mamur Shuhratov

Assistant of the Department of Artificial Intelligence

Elyor Egamberdiev

Assistant of the Department of Artificial Intelligence

Ziyoda Norqulova

Assistant of the Department of Artificial Intelligence

THE ROLE AND APPLICATION OF ARTIFICIAL INTELLIGENCE IN IDENTIFYING THREATS TO INFORMATION SYSTEMS

Bahodir Muminov,
Tashkent State University of Economics,
Uzbekistan

Obidjon Bekmirzaev,
Tashkent State University of Economics,
Uzbekistan,
bekmirzayevobidjon1989@gmail.com

A B S T R A C T **In this article, In recent years, artificial intelligence has become a necessary technology to enhance the efforts of information security professionals. From a security point of view, it is important that artificial intelligence can identify and prioritize risks, immediately detect any malware on the network, respond to incidents and detect attacks in advance, and develop algorithms to detect and block botnets in computer networks.**

K E Y W O R D S Information system, artificial intelligence, information security, cyber security, attack, threat, risk, method, model, botnet, IDS, IPS, Botminer, BotSniffer, BotHunter, N-gram, Tight, Sustainability, Incremental, FluXor, Tamed, Markov and Synchronization.

INTRODUCTION

Today, information technology plays an important role in all aspects of our life. At this point, it can be said that users widely use computers, smartphones and other digital devices in their daily life, work and study. At the same time, these technologies carry a great responsibility in protecting users' personal information and financial resources. Cybersecurity is very important in this regard.

In this technologically advanced age, Cybersecurity analysis and improvement is no longer the responsibility of humans alone. In response to this unprecedented challenge, artificial intelligence (AI)-based cybersecurity tools have emerged to help information security teams reduce the risk of hacking and increase security. [1-3].

METHODS

Artificial intelligence and machine learning (Machine Learning) have become the most important technologies in the field of information security, because they have the ability to quickly analyze millions of events and identify various threats. Able to detect malicious behavior that can lead to phishing attacks or malicious code uploads, from malware exploiting zero-day vulnerabilities. These technologies build on past experiences and are now evolving over time to detect new types of attacks.

Artificial intelligence and data analysis

Unfortunately, artificial intelligence is a very popular but often misused buzzword these days. Currently, many companies are looking for ways to move to the side of artificial intelligence. But many AI proposals today don't actually meet the requirements of AI testing. Although they use technology that analyzes data and allows certain results to be determined, it is not AI. Pure AI consists of augmenting cognitive abilities to automate tasks.

Artificial intelligence systems are iterative and dynamic. As they analyze more data, they become smarter, "learns" from experience, and become increasingly capable and autonomous.

Data analytics, on the other hand, is a static process that uses specialized systems and software to examine large sets of data to make inferences about the information in them.

Machine learning, expert systems, neural networks, and deep learning are examples of artificial intelligence technologies:

Machine learning – works well only when it is focused on solving a specific task and not on a large-scale mission;

Expert systems – are programs designed to solve problems in specialized fields. These systems mimic the thinking of real experts, solving problems and making decisions through fuzzy rule-based reasoning using carefully selected knowledge sets;

Neural networks use a biological programming paradigm that allows a computer to learn from observational data.

Deep learning is a part of machine learning methods based on providing training data, as opposed to algorithms specific to a specific task. Today, image recognition through deep learning is often used in fields such as autonomous vehicles, scan analysis, and medical diagnostics.

Application of artificial intelligence in cyber security

Artificial intelligence can help solve some of the toughest cybersecurity challenges. With today's ever-evolving cyberattacks and device proliferation, machine learning and artificial intelligence can be used to "track the bad guys" by automating threat detection and responding to threats more effectively than traditional software-based approaches.

However, cyber security comes with a number of unique challenges:

- Huge attack area;
- 10 or 100 thousand devices in each organization;
- Hundreds of attack vectors;
- Lack of qualified security professionals.

An automatic artificial intelligence-based cyber security situation management system is able to solve many of these problems.

Here are the different levels of artificial intelligence:

- - Inventory of information technology assets;
- Exposing threats;
- Effective control
- Predicting the risk of hacking
- Responding to incidents.

RESULTS

Let's consider botnets from attacks aimed at information systems

Top Ways to Prevent Botnet Attacks: Given that a botnet attack is difficult to detect in time and the consequences of a breach can be significant, here are the top preventative measures you can take to protect your organization from botnet attacks.

The fight against botnets is a complex and multifaceted task that requires the joint efforts of security experts, law enforcement agencies, Internet service providers and users.

The following methods can be used to combat botnets:

Botnet detection. To detect a botnet, network activity and computer behavior must be analyzed for signs of infection. For example, you can use antivirus software, network scanners, intrusion detection systems (IDS), and intrusion prevention systems (IPS).

In order to prevent bots from infecting computers, it is necessary to observe the rules of cyber hygiene, for example: use reliable antivirus programs and update them regularly, install patches and updates for the operating system and applications, do not open suspicious attachments and links, do not use pirated programs and downloadable tools , do not grant administrative rights to unverified programs, etc.

Analysis of "Botnet" detection algorithms in computer networks

In order to analyze algorithms for detecting botnets in computer networks, we give a brief description of these algorithms.

BotSniffer can capture spatio-temporal correlations in network traffic and use statistical algorithms to detect botnets with theoretical limits on false positive and false negative rates. BotHunter is not an intrusion detection system, firewall, spam blocker or antivirus tool. These tools usually won't work to rid your network of malware infections. In contrast, BotHunter takes a different approach. It's a network protection tool designed to help everyone from network administrators to Internet-connected PC users detect when coordination-based malware (such as botnets, spam bots, spyware, trojan exploiters, worms, adware) is running on their systems. is a completely new algorithm. BotHunter monitors the two-way traffic between your internal network and hosts on the Internet. We analyzed the algorithms in the table below.

It compares three features: check, define and compare. Verify means to verify the data set. Pre-processing refers to pre-processing a data set. Benchmarking means comparing the results with the results of other offers. Each feature can be fully developed (~), moderately developed (+), or not developed at all (-). So you can quickly compare before reading the description. The analysis text of each proposal describes these features in detail.

TABLE 1.Analysis of botnet detection algorithms

Name	Check	Identify	Comparison
Botminer	+	+	-
BotSniffer	+	+	-
BotHunter	+	-	-
N-gram	-	+	-
Tight	+	~	-
Sustainability	+	~	-
Incremental	-	+	-
Model	+	~	-
FluXor	~	+	-
Tamed	+	+	-
Markov	+	~	-
Synchronization	~	~	-

BotSniffer The first two detection approaches group together hosts that connect to the same remote server and port and sniff network packets with a common baseline, and separate these groups into time windows. With this information, the first approach looks for fingerprints (such as sending SPAM, binary downloads, and port scanning) of the hosts that caused the attack. If more than half of the group has carried out attacks, then the group is defined as a possible botnet. The sequential likelihood ratio test algorithm is used to determine if it is a botnet.

The BotMiner detection system has three different stages of analysis. The first stage unites hosts with similar forms of activity. The second phase groups hosts with similar attack patterns, and the third phase groups hosts based on similarities between the other two phases. The first step uses flow information such as Internet Protocol (IP) addresses, ports, etc. from the network profile. calculates statistics such as fl loan per hour, packets per stream, average bytes per packet, and average bytes per second. A two-step X-means clustering method is used to create clusters of hosts that behave similarly in a network. The second phase uses Snort IDS to extract attacks from each host and group hosts by attack type. Malicious activities detected include SPAM, exploit attempts, port scanning, and binary downloads. The third step calculates a score for each attacking host based on the previous similarities and uses it to calculate a similarity function between the hosts. Finally, a dendrogram is generated using this function to determine the best cluster separation. The most important static properties used were mail exchange DNS queries for SPAM detection and TCP port 25 for SMTP. The proposal uses both virtualized and real botnet images. Unfortunately, there is no information on how the botnet dataset was verified. Normal captures were checked using BotHunter and BotSniffer. Some well-known websites are whitelisted during the pre-processing stage. However, the effects of the filter are not disclosed. This also makes maintaining a list of known hosts, which is error-prone and time-consuming. The results look encouraging. However, the data set could not be reproduced because it is not publicly available. No comparison with other papers was made. Unlike other proposals, this work uses one new idea to distinguish between botnets and manual attacks: botnets act maliciously and always communicate in a similar way, but manual attacks only work with malicious intent.

The BotHunter system recognizes the infection and coordination dialog of botnets by matching a state-based infection sequence model. It captures packets as they exit the network using a modified version of Snort IDS with two proprietary detection plugins. The proposal looks for evidence of botnet life cycle stages to analyze bot dialogue correlations. IDS alerts are tracked through a temporary window and contribute to each host's infection score. When certain thresholds are exceeded, the host is marked as a bot. This proposal associates incoming scans and intrusion alerts with outgoing communication patterns.

The N-gram work proposes an unsupervised, classification-based, and IRC-based bot traffic detection method. It consists of two steps: application traffic classification and bot detection. The first stage consists of several stages. The first phase classifies traffic to specific applications using signature-based payloads and specific ports (the signatures were previously generated). The second stage classifies the unknown traffic from the previous stage using a decision tree based on temporal-frequency characteristics. This tree differentiates certain communities of applications (proposals have temporal-frequency properties of certain flows). In the third step, an anomaly-based approach is used for traffic clustering.

The Tight proposal aggregates traffic to identify hosts that may be part of a botnet.

The training dataset consists of 600 GB of normal wireless traffic from the Crawdad archive and 74 streams of the botnet constructed from an internal testbed. The proposed method consists of five steps. First, the streams are shuffled in the shared data set (unfortunately, there is no information on how the shuffle is done). Second, whitelisting and blacklisting are used to filter streams. Third, streams are classified according to their conversation-like characteristics. Fourth, correlations are used to find similar C&C streams. Fifth, a topological analysis of the flows is carried out to identify common connection points and manually identify the traffic between the controller and the controller. meeting point. The paper makes several assumptions: first, the IRC-based botnet command messages sent by the botmaster are short and text-based; second, two different streams of the same application behave similarly; and thirdly, botnet C&C channels do not provide bulk data transfer. The analysis shows that some filters are better suited to the data, such as removing high bitrate streams. However, these filters have not been tested and appear to be specifically designed to filter out already known botnet traffic. There is a problem with the dataset. When botnet packets were captured, secure shell traffic from the testbed installation was also captured. This will break the capture. Almost every dataset has similarities, but it's important to list and review them. Also, the regular shots have not been confirmed and the data set has not been published. There is another bias in the results. Some results were obtained by applying the classifiers to a single data set; Other results were obtained by applying the classifiers to other datasets. There should be at least three types of data sets generally accepted: training, testing, and validation. Evaluation methodology should also be used. The design of the proposal is believed to be determined in real time, but unfortunately no experiments have been conducted to support this idea. The main point of the article is that it focuses on new behavioral features: the bot communicates with the C&C server, the server is controlled by the botmaster, and the bots synchronize their actions.

In stability, P2P botnets are detected using flow collection and stability metrics. The dataset was created by capturing background traffic on a university campus and then injecting real botnet traffic into it. Botnet traffic is captured in a honeynet and is therefore proven to be malicious by the definition of a honeypot.

Incremental botnet activity is controlled based on traffic similarity between characteristic flows of multiple hosts. The proposal is divided into five stages. The first step captures normal and botnet traffic from the internal network. P2P botnet traffic is received by a third party. The snapshots at this stage were simulated because the internal network did not have access to the Internet. The second step reduces the size of the simple data by applying some filters. Non-TCP files are filtered, several whitelists and blacklists of public websites are applied, and packets larger than 300 bytes are dropped. The third step creates characteristic flows. The traffic is divided into 5-minute time windows and two characteristics are calculated: the average number of bytes per packet and the number of packets. In the fourth step, the measure of the distance between the streams is calculated using the incremental discrete Fourier transform technique. A new feature of this technique is that it is not necessary to recalculate all the coefficients every time a new value arrives. In the fifth step, Snort IDS is used to manually inspect the activity of suspicious hosts (hosts with high similarity streams). This step also decides if the host is part of a botnet or not. Unfortunately, there is no information about this step. The proposal assumes that simulated botnet traffic looks like real botnet traffic, and that the traffic from their network is normal. The proposal is not designed to differentiate between botnet and other attacks. Analysis shows that the results are inconclusive. None were calculated and excellent detection accuracy was reported. The paper also stated that "the false positive rate is still relatively low because we will analyze their performance to confirm the final result. Unfortunately, the confirmation is not in the proposal. The results cannot be compared with other papers. The proposal is pre-processed uses several filters in the giving phase. However, the bias added by these filters is used in the proposal. However, there is no information about the distance measurement and start time of the protocol First, the method is difficult to replicate.The process of the mixture is not explained.It is capable of tracking botnets in real time Second, feature streaming, while not new, is an important improvement in the field.

The models proposal detects single virus machines using previously created detection models. These models consist of two parts. The first part extracts the strings received by the bot on the network to find the commands. The second part tries to find responses (attacks) to the sent command. The models are saved for later use in real networks. The new idea is to look for command signatures (string tokens) in network traffic and then look for responses to those switches.

FluXor's offering identifies and monitors high-flow domains. Given a fully qualified domain name, its purpose is to verify that it is masking an express service network and to identify all agents that are part of the network in such a case. Although not a botnet detection method, this is one of the few suggestions that can identify fast-flowing botnet domains. Suspicious domains are identified when traces of fast flow properties are found in a domain's DNS and WHOIS data. Fast-flow traffic is determined by nine attributes that describe domain characteristics, such as the degree of availability and

the heterogeneity of hosts in DNS A records. The experimental setup includes a collector module that collects domains from various sources, a monitor module that tracks domains, and a detector module that provides a classification algorithm from the Waikato Environment for Knowledge Analysis suite. This proposal does not have a detailed performance analysis, so it is not possible to assess how well it has performed. The analysis shows that malware domains are obtained from SPAM emails, while normal domains are obtained from web history. However, there is no description of domain validation. Unfortunately, due to privacy and security concerns with domain names, the dataset has not been made public. On the other hand, the algorithm may be replicable because the Waikato Environment for Knowledge Analysis framework is publicly available.

Tamed's proposal hypothesizes that "even stealthy, previously unseen malware can exhibit detectable communication". The proposal is to "identify infected internal hosts through communication discovery, aggregates that consist of flows with common network characteristics. This is the first proposal that does not attempt to detect botnets themselves, providing a consistent and useful set of features that future work can rely on to detect them. The work is divided into two stages: determination of their cumulative feature function and evaluation of experiments. In the first step, three different aggregation functions are defined: the target aggregation function, the payload aggregation function, and the common platform function. The Destination Aggregate feature finds suspicious destination subnets that have a large number of interactions with the internal network (and IP addresses involved). This is done by comparing the previous network entry with the current network entry. The function consists of two steps. First, past normal traffic is used to remove periodic connections. Second, a principal component analysis algorithm is used to find the most important components, and then a clustering method is used to find hosts connecting to the same address combinations. The payload uses the summation function edit distance with substring moves to output the normalized ratio. This ratio indicates how much distance between two payloads is below a certain threshold (and therefore needs to be trained). The proposal also includes an algorithm for approximating the fraction of pairs of related records that satisfy this distance (using the nearest-neighbor option). The common platform function uses two heuristics to passively fingerprint host operating systems: time fields and communication properties (for example, Windows computers connected to Microsoft Time Server). This last function returns the largest number of internal hosts that can be identified with the same operating system. The suggestion is that the C&C server may be on a different network, and it is assumed that the bots are not using the Tor.* network. The analysis shows that the normality of the traffic recorded at the university is not confirmed, but given as normal. In one experiment, traffic from one botnet was combined with traffic from a university. The IP addresses of the bots were replaced with the IP addresses of the hosts at the university.

Markov: Detected in port scanning activity of botnets. The scanning steps are represented by text symbols and a hidden Markov model (HMM) is used for training and detection. The work is divided into four stages. In the first step, the system is used to retrieve information from the network and create network logs. Unfortunately, this system is not illustrated. The data was collected in 11 C-category university networks for 1 month. By the way, an unknown number of hosts were found to be infected. In the second step, the number of TCP packets with SYN and ACK bits present in certain time windows is calculated. Unfortunately, no information about time windows has been provided. The amount of packets is plotted and a different text symbol is assigned to each individual form of traffic. This step ends with a string of letters (one gram) representing each port scan (observation). In the third step, the training step, the Baum-Welch algorithm is used to find the Markov model that maximizes the likelihood of each sequence of observations over 6 hours. In the fourth step, strings are extracted from the rest of the traffic (presumably of an unknown type). However, not much is known about it. Next, the Viterbi algorithm is used to find the most probable sequence of states in the HMM from each group of observed states, that is, to find the model that best explains the observations. The result of Viterbi algorithms is used as a type of score.

Synchronization: Used to detect bots in network synchronization and clustering methods. There are five stages. At the first stage, data is collected. Three different test beds were used to obtain five sets of data. Three of the datasets correspond to botnet images and two to non-botnet images. Non-botnet images include manual port scans and normal traffic. Validation of captures is performed for both botnet and non-botnet datasets. Non-bot traffic is verified by the operating system, antivirus software, and a fresh installation of Snort IDS. In the second step, TCP streams are extracted using the tcptrace tool. In the third step, the flows are divided into 1-second time windows and three properties are used to aggregate them: the amount of unique source IP addresses in the time window, the amount of unique destination IP addresses in the time window, and the amount. single destination ports in a time window. Each instance represents a time window with aggregated properties. At the fourth stage, the expectation-maximization algorithm is used for clustering examples. A dataset with both types of traffic was obtained by combining botnet and non-botnet data. Four experiments were conducted: first, to distinguish between botnets and port scanning activities; second, separating botnet and non-botnet traffic from a single host; third, distinguish between botnet and non-botnet traffic in an unbalanced dataset; and fourth, separating the network of bot and non-botnet hosts.

Development of an algorithm for identifying and blocking "botnets" in computer networks.

We need to develop an algorithm for detecting and blocking "botnets" in computer networks. This requires detection of unwanted activities in networks, data analysis and security measures. Below is a general description and steps of the algorithm for detecting and blocking botnets.

Botnet *Detection* and *Blocking* Algorithm

1. Data collection. Collection of network traffic data:

- IP addresses;
- Ports;
- Protocols;
- Timestamps of network packets;
- URL- addresses.

2. Preprocessing. Network data cleaning and preprocessing:

- Noise removal;
- Normalization;
- Division by time.

3. Feature extraction. Extracting important features for identification:

- Number of packages;
- Package size;
- Relationship between IP addresses and ports;
- Average traffic speed;
- Number of unique IP addresses.

4. Model training. training the model in the botnet blocking algorithm.

5. Detect botnets:

- Detection of botnets using a machine learning model;
- Classification algorithms (Random Forest, SVM, Neural Networks, etc.).
- *6. Block botnets. Block detected botnets*:
- Updating firewall rules;
- Blacklisting of IP addresses;
- Restricting network access.

We have created an algorithm, now we will consider the software part. I wrote the software part in the Python programming language.

Real network data is required for the program to work fully. But for example, it is possible to test the code using artificial data. Below we will create artificial data.

We generate artificial network data, which includes both botnet and normal traffic.

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score
# 1. Creation of artificial data
def create_synthetic_data():
    np.random.seed(42)
    # 1000 normal traffic and 100 botnet traffic
    normal_traffic = pd.DataFrame({
        'packet_size': np.random.randint(20, 1000, 1000),
        'packet_count': np.random.randint(1, 100, 1000),
        'unique_ips': np.random.randint(1, 50, 1000),
        'avg_packet_rate': np.random.uniform(0.1, 2.0, 1000),
        'duration': np.random.uniform(0.1, 10.0, 1000),
        'protocol': np.random.choice(['TCP', 'UDP', 'ICMP'], 1000),
        'is_botnet': 0,
        'ip_address': np.random.choice([f'192.168.1.{i}' for i in range(1, 255)], 1000)
    })
    botnet_traffic = pd.DataFrame({
        'packet_size': np.random.randint(500, 1500, 100),
        'packet_count': np.random.randint(50, 200, 100),
        'unique_ips': np.random.randint(20, 100, 100),
        'avg_packet_rate': np.random.uniform(1.0, 5.0, 100),
        'duration': np.random.uniform(5.0, 20.0, 100),
        'protocol': np.random.choice(['TCP', 'UDP', 'ICMP'], 100),
        'is_botnet': 1,
        'ip_address': np.random.choice([f'10.0.0.{i}' for i in range(1, 100)], 100)
    })
    data = pd.concat([normal_traffic, botnet_traffic], ignore_index=True)
    return data
```

```
['10.0.0.32' '10.0.0.67' '10.0.0.3' '10.0.0.86' '10.0.0.74' '10.0.0.51'
 '10.0.0.76' '10.0.0.93' '10.0.0.48' '10.0.0.17' '10.0.0.7' '10.0.0.13'
 '10.0.0.66' '10.0.0.4' '10.0.0.10' '10.0.0.88' '10.0.0.52' '10.0.0.6'
 '10.0.0.83' '10.0.0.69' '10.0.0.96' '10.0.0.9' '10.0.0.91' '10.0.0.77'
 '10.0.0.21' '10.0.0.24' '10.0.0.59' '10.0.0.99' '10.0.0.68' '10.0.0.40'
 '10.0.0.54' '10.0.0.35' '10.0.0.95' '10.0.0.56' '10.0.0.89' '10.0.0.46'
 '10.0.0.78' '10.0.0.27' '10.0.0.36' '10.0.0.81' '10.0.0.2' '10.0.0.92'
 '10.0.0.43' '10.0.0.90' '10.0.0.22' '10.0.0.55' '10.0.0.60' '10.0.0.58'
 '10.0.0.12' '10.0.0.84' '10.0.0.39' '10.0.0.41' '10.0.0.57' '10.0.0.53'
 '10.0.0.97' '10.0.0.80' '10.0.0.45' '10.0.0.5' '10.0.0.11' '10.0.0.71'
 '10.0.0.34' '10.0.0.72' '10.0.0.31' '10.0.0.75']
```

```
# 2. Preprocessing
def preprocess_data(data):
    data = data.dropna()
    data['timestamp'] = pd.to_datetime('now')
    data['protocol'] = data['protocol'].map({'TCP': 0, 'UDP': 1, 'ICMP': 2})  return data
# 3. Feature extraction
def extract_features(data):
    features = data[['packet_size', 'packet_count', 'unique_ips', 'avg_packet_rate', 'duration', 'protocol']]
    labels = data['is_botnet']
    return features, labels
# 4. Model training
def train_model(features, labels):
    X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.2, random_state=42)
```

```
    model = RandomForestClassifier(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(classification_report(y_test, y_pred))
    print("Accuracy:", accuracy_score(y_test, y_pred))
    return model
# 5. Detect botnets
def detect_botnets(data, model):
    features = data[['packet_size', 'packet_count', 'unique_ips', 'avg_packet_rate', 'duration', 'protocol']]
    predictions = model.predict(features)
    botnet_ips = data[predictions == 1]['ip_address'].unique()
    return botnet_ips
# 6. Botnetlarni bloklash
def block_botnet_ips(botnet_ips):
    print("Botnet IPs to be blocked:", botnet_ips)
    return botnet_ips
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	204
1	1.00	1.00	1.00	16
accuracy			1.00	220
macro avg	1.00	1.00	1.00	220
weighted avg	1.00	1.00	1.00	220

Fig. 1. Classification Report

The model was trained and tested. Below are the accuracy reports and detected botnet IP addresses:

Fig. 2. Botnet IP Addresses Expected to be Blocked*Aniqlik*

DISCUSSION

The above results show that the model can detect botnet IP addresses with very high accuracy. Using this information, it is possible to block identified botnet IP addresses.

To work in a real network environment, you need to collect real network data and perform this process in real time.

CONCLUSION

In recent years, artificial intelligence has become a necessary technology to enhance the efforts of information security professionals. In terms of security, artificial intelligence has been developed to identify and prioritize risks, immediately detect any malware on the network, respond to incidents and detect attacks in advance, and develop an algorithm to detect and block botnets in computer networks.

References

[1] Nuralievich, B. O., & Boltaevich, M. B. (2021, November). Method of Detection and Elimination of Tracks of Attacks in the Information System. In 2021 International Conference on Information Science and Communications Technologies (ICISCT) (pp. 1-2). IEEE.

[2] Nuralievich, B. O., Boltaevich, M. B., & Ugli, B. U. B. (2022, September). The Procedure for Forming a List of Sources of Attack in the Information System. In 2022 International Conference on Information Science and Communications Technologies (ICISCT) (pp. 1-4). IEEE.

[3] Bekmirzaev O., Shirinov B. An Algorithm for Viewing Node State Events Under Attack for Information Systems // AIP Conference Proceedings., 2024, 3147(1), 050003. DOI: 10.1063/5.0210404

[4] Bekmirzaev O., Samarov H. A Method of Evaluating the Effectiveness of Information System Protection // AIP Conference Proceedings., 2024, 3147(1), 050004. DOI: 10.1063/5.0210405

[5] Muminov, B., & Bekmirzaev, O. (2022). Classification and analysis of network attacks in the category of "denial of service" information system. central asian journal of education and computer sciences (CAJECS), 1(1), 7-15.

[6] Axmedova, N., & Bekmirzaev, O. (2022). Analysis of methods of fighting against network attacks of the "denial of service" category on information systems. central asian journal of education and computer sciences (CAJECS), 1(5), 17-23.

[7] Muminov, B., & Bekmirzaev, O. (2022). Structure and algorithms of online discussion information system. Scientific Collection «InterConf», (114), 373-384.

[8] Мўминов, Б., & Бекмирзаев, О. (2022). Построение узлов о событиях под влиянием атаки в информационной системе. Scientific Collection «InterConf», (114), 388-396.

[9] Bekmurodov, O. (2023). Ахборот тизимларида ҳужум манбалари рўйхатини шакллантириш процедураси. Digital Transformation and Artificial Intelligence, 1(3), 129-136.

[10] Samarov, H. K., & Bekmirzayev, O. N. (2023). Masofaviy o'qitish tizimlarida mavjud risklar va ularni minimallashtirish istiqbollari. Research and Education, 2(4), 146-155.

[11] Bekmirzayeva, M. (2024). VIZUALLASHTIRISH TIZIMLARIDA YOMG 'IR VA QOR MUAMMOLARINI TASVIRGA DASTLABGI ISHLOV BERISH YORDAMIDA BARTARAF ETISH. DIGITAL TRANSFORMATION AND ARTIFICIAL INTELLIGENCE, 2(1), 120-124.

[12] Bekmirzayev, O., & Sabirov, X. (2023). "KOMPYUTER ARXITEKTURASI" FANINI O 'QITISH SAMARADORLIGINI OSHIRISHDA ZAMONAVIY PEDAGOGIK TEXNOLOGIYALARNING INTERFAOL USULLARIDAN FOYDALANISH. Science and innovation, 2(Special Issue 14), 549-551.