УДК 004.056
DOI: 10.26102/2310-6018/2025.51.4.028

# Phishing link detection system based on explainable AI technologies

**A.F. Shaimardanov[1], A.M. Vulfin[1,2]✉, A.D. Kirillova[1], A.V. Minko[1]**

[1]*Ufa University of Science and Technology, Ufa, the Russian Federation*
[2]*Omsk State Technical University, Omsk, the Russian Federation*

***Abstract.*** A set of models for analyzing symbolic domain names in the tasks of detecting phishing links has been developed based on the construction of an ensemble of classifiers that are optimized for hardware platforms. This allows for increased efficiency of analysis when integrated into existing information security operation centers. The results of testing on real data for key metrics confirm the high accuracy of detecting malicious links. Software with a microservice architecture has been developed for integration into the information system of the security operation center. The proposed models are optimized for use on CPU by translating them into compiled code, which increased the computational performance of the models by 26 %. Classifier models based on the Code-BERT transformer, retrained on a prepared data set, are proposed. Modules of the subsystem for explaining the decision taken have been developed using methods of explainable artificial intelligence – the use of techniques for composing a query for a locally deployed large language model with a description of the signs of malicious links using zero-shot learning.

***Keywords:*** machine learning, phishing, phishing link detection system, security operation center, explainable artificial intelligence, large language model.

***Acknowledgements:*** The research was carried out by Omsk State Technical University within the framework of the state assignment of the Ministry of Science and Higher Education of the Russian Federation for 2023-2025 No. FSGF-2023-0004.

***For citation:*** Shaimardanov A.F., Vulfin A.M., Kirillova A.D., Minko A.V. Phishing link detection system based on explainable AI technologies. *Modeling, Optimization and Information Technology*. 2025;13(4). URL: https://moitvivt.ru/ru/journal/pdf?id=2066 DOI: 10.26102/2310-6018/2025.41.4.028

# Система обнаружения фишинговых ссылок на основе объяснимых технологий искусственного интеллекта

**А.Ф. Шаймарданов[1], А.М. Вульфин[1,2]✉, А.Д. Кириллова[1], А.В. Минко[1]**

[1]*Уфимский университет науки и технологий, Уфа, Российская Федерация*
[2]*Омский государственный технический университет, Омск, Российская Федерация*

***Резюме.*** Разработан комплекс моделей анализа символьного доменного имени в задачах обнаружения фишинговых ссылок, на основе построения ансамбля классификаторов, отличающихся оптимизацией для аппаратных платформ, что позволяет повысить оперативность анализа при встраивании в существующие системы мониторинга информационной безопасности. Результаты тестирования на натурных данных по ключевым метрикам подтверждают высокую точность обнаружения вредоносных ссылок. Разработано программное обеспечение с микросервисной архитектурой для интеграции в информационную систему центра мониторинга информационной безопасности. Предложенные модели оптимизированы для использования на центральном процессоре путем перевода их в скомпилированный код, что увеличило вычислительную производительность моделей на 26 %. Предложены модели классификаторов на основе трансформера Code-BERT, дообученного на подготовленном наборе данных. Разработаны модули подсистемы объяснения принимаемого решения с помощью методов объяснимого искусственного интеллекта – применения техник составления запроса для локально развернутой большой языковой модели с описанием признаков вредоносных ссылок zero-shot learning.

***Ключевые слова:*** машинное обучение, фишинг, система обнаружения фишинговых ссылок, центр мониторинга информационной безопасности, объяснимый искусственный интеллект, большая языковая модель.

***Благодарности:*** Работа выполнена ОмГТУ в рамках государственного задания Минобрнауки России на 2023-2025 годы № FSGF-2023-0004.

***Для цитирования:*** Шаймарданов А.Ф., Вульфин А.М., Кириллова А.Д., Минко А.В. Система обнаружения фишинговых ссылок на основе объяснимых технологий искусственного интеллекта. *Моделирование, оптимизация и информационные технологии.* 2025;13(4). (На англ.). URL: https://moitvivt.ru/ru/journal/pdf?id=2066 DOI: 10.26102/2310-6018/2025.41.4.028

## Introduction

The number of attacks on information systems using social engineering methods is constantly increasing, including attacks with the substitution of links to official websites of organizations (phishing) to obtain confidential information. The use of organizational measures and user training should be accompanied by the use of built-in and imposed means of protecting information from targeted and fan-out phishing attacks. Software implementation of models and algorithms for detecting phishing links will improve the efficiency (speed) of analysis and reduce the workload of monitoring specialists.

Detection of phishing links is based on an assessment of compliance with URL formation standards and a number of heuristic rules related to symbolic filtering and analysis of information about the domain name of the target node. The digital space is evolving rapidly, and phishing methods are becoming more sophisticated. Traditional filters based on static rules are no longer sufficient as attackers dynamically change domains and replace symbols [1]. Heuristic rules generally use a limited set of features and patterns that are difficult to keep up to date. Machine learning (ML) methods are capable of using a significantly larger number of features, such as WHOIS information about a domain, the status of SSL certificates, etc.

The use of machine learning methods [2] will allow, in conditions of fuzzy initial data, to increase the probability of detecting phishing links [3, 4] and the speed of their analysis due to the software implementation of the machine learning module. Models based on neural networks are actively used: convolutional neural networks [5], transformer models [6]. However, the performance of such models in non-batch classification mode (processing small batches of queries or single queries) is significantly lower than that of classical decision tree-based machine learning models. The improvement in classification quality (reduction in type II errors) is insignificant. It is promising to use such models to explain the decision being made and subsequent retrospective analysis of the work of traditional models for the purpose of their further training.

The purpose of the work is to increase the efficiency (speed) of detecting phishing links through the development and software implementation of machine learning models and algorithms for classifying character sequences.

## Materials and methods

**Classical machine learning technologies and explainable artificial intelligence technologies in the problem of detecting phishing links.** The website address (URL/URI) includes the protocol, domain name (with subdomains), path and request parameters. The domain name and overall link structure are of particular importance for detecting phishing.

Attackers create URLs that are as similar as possible to legitimate ones, or complicate them by adding extra characters, multiple subdomains, numbers within words, or replacing letters with similar ones from a different alphabet in order to bypass standard filters and confuse users and automated systems [7].

To detect phishing links, the system analyzes the URL, highlighting various features [8]. Among them are statistical characteristics – the total length of the URL, the length of the domain name, the number of subdomains, the number of unique characters, the proportion of digits and the level of entropy. Additionally, suspicious substrings such as “secure”, “login”, “verify”, as well as substitution characters (for example, the Cyrillic “р” instead of the Latin “p”) are checked.

For deeper analysis, linguistic features are introduced through N-gram analysis, where the domain name is represented as a sequence of substrings (bi- and trigrams, etc.), and their significance is determined using TF-IDF. An additional set of test words helps identify hidden patterns even in previously unseen URLs. This comprehensive approach enables the creation of a multidimensional feature space that evaluates URLs from different angles – from quantitative characteristics to text patterns.

The use of phishing link detection systems is associated not only with issues of ensuring performance, but also with the need to explain the decision being made. Explainable artificial intelligence (XAI) [9] technologies (Table 1) include methods and tools that allow the results of a classifier related to black-box models (primarily neural network models) to be interpreted in a human-readable format (e.g., a set of rules) and to increase the transparency of the solution by assessing the significance of features both at the level of individual examples and at the level of the classifier as a whole (Table 2).

Table 1 – Applied technologies and algorithms of XAI
Таблица 1 – Применяемые технологии и алгоритмы объяснимого искусственного интеллекта

| Category | Examples | Description |
|---|---|---|
| Local | LIME, SHAP, Counterfactuals | Explain the solution for one example:<br>– LIME builds a local linear model in the vicinity of one example;<br>– SHAP calculates the contribution of each feature (Shapley values);<br>– Counterfactuals take into account “minimal” changes in feature values. |
| Global | Feature Importance, Partial Dependence Plots | Allows to identify general patterns:<br>– ranking of features by importance;<br>dependence of the forecast on one feature. |
| Integrated | Attention mechanisms, Saliency Maps | Internal features of models (e.g. visual):<br>– attention weights in transformers;<br>– gradient heat maps (Grad CAM). |
| Conceptual | TCAV (Testing with Concept Activation Vectors) | Human-readable explanation using concept map. |

Table 2 – Using deep neural network models in the tasks of detecting phishing links and explaining the decision made
Таблица 2 – Использование глубоких нейросетевых моделей в задачах выявления фишинговых ссылок и объяснения принимаемого решения

| Type | Openness / accessibility | Model | Architecture, openness and training data | Purpose | Presence and mechanism of explanation |
|---|---|---|---|---|---|
| General Purpose LLM | Closed model, available via API | GPT-4 (OpenAI), 2023 | LLM approximately 175B parameters; trained on an array of prepared text data from the Internet. | A universal model; applicable to URL classification via zero-shot or few-shot. | Yes – building a chain-of-thought based on a specialized request (prompt). |
| | | Claude 3 (Anthropic), 2025 | LLM approximately 100B parameters; trained on an array of prepared text data from the Internet. | | Yes – can explain the solution in the answer to the instructions (instructions for explanation). |
| | Open-source, can be deployed locally and is accessible via API. | Llama 2 (Meta), 2023 | LLM 7–70B parameters. | Universal model; requires fine-tuning (LORA) or a detailed query (prompt) to analyze the URL. | Partially – without additional training the quality of explanations decreases, but when using examples in the prompt it will try to justify the answer. |
| BERT/Trans-former based models | Open-source, can be deployed locally and is accessible via API. | DomURLs_BERT (2024) [10] | Pre-trained BERT encoder (base) on 395M URLs (symbols, lexemes, DGA domains) with subsequent additional training on a labeled set of malicious and safe URLs. | Binary and multi-class classification of domains and URLs (phishing, malware, DGA, etc.). | Not directly (class label only). It is possible to analyze Attention or Grad-CAM layers to identify parts of the URL that are considered malicious by the model, as well as to use external methods (LIME, SHAP) of analysis. |

Table 2 (continued)
Таблица 2 (продолжение)

| | | | | | |
|---|---|---|---|---|---|
| | | CodeBERT-base-malicious-urls[1] (2022) | Pre-trained CodeBERT with fine-tuning on a sample of malicious / benign URLs. | Binary URL classifier (malicious / benign); improved accuracy by paying attention to URL structure at the character level. | |
| | | URLTran (2021) [11] | BERT transformer trained on a corpus of phishing URLs. | | |
| Hybrid models | | BERT-CNN (2023) [12] | URL classification model based on a combination of the pre-training BERT model and the convolutional neural network (CNN). | Binary and multi-class classification of domains and URLs (phishing, malware, DGA, etc.). | |
| Deep neural networks | | URLNet (2018) [13] | Deep CNN network trained on character and dictionary URL embeddings. | Multi-class URL classification (phishing, spam, exploit) based on selected lexical features and attributes at the symbol level. | |
| ML models | Open-source, can be deployed locally and is accessible via API. | CatchPhish (2020) [14] | A random forest classifier with a set of hand-selected URL features (e.g. presence of @, domain length). | Binary and multi-class classification of domains and URLs (phishing, malware, DGA, etc.). | No (class label only). Analysis using external methods (LIME, SHAP) is possible. |
| | | PhishRF (2019) [15] | Random Forest Classifier with Hand-Choosed URL Features. | | |
| External and local services | Closed model, available via API | VirusTotal | Ensemble of signatures and models, closed algorithms. | URL analysis for threats (phishing) using signature databases and the use of ML models. | Usually no – they give out a category or characteristics, but without a textual explanation of the reasons. |

[1] DunnBC22. NLP_Projects / Multiclass Classification / Malicious URLs. GitHub. URL: https://github.com/DunnBC22/NLP_Projects/tree/main/Multiclass%20Classification/Malicious%20URLs [Accessed 29th August 2025].

Table 2 (continued)

Таблица 2 (продолжение)

| | | | | | |
|---|---|---|---|---|---|
| | Based on an Open-source model, available locally[2] | LLM Guard | Pre-trained CodeBERT with fine-tuning on a sample of malicious / benign URLs. | Binary URL classifier (malicious / benign); improved accuracy by paying attention to URL structure at the character level. | No. The model provides a score between 0 and 1 for a URL being malware. Analysis using external methods (LIME, SHAP) is possible. |
| As part of application software | Closed model | Corporate products: email clients (Outlook, Gmail) | – | – | Usually no – they give out a category or characteristics, but without a textual explanation of the reasons. |

Researchers [16] aim to make threat detection explainable to increase user confidence and help analysts understand why a detection occurred. Let's look at approaches that allow LLM and other AI models to explain why a URL is classified as malicious and what type of attack it represents.

General-purpose LLM models can be nearly as accurate as specialized URL classifiers if the query (prompt) is formulated correctly and in detail, and are also capable of explaining the solution in natural language. In [16], the best results of LLM GPT-4 Turbo and Claude 3 achieved an F1-score of 0.92 in the zero-shot prompt mode.

Transformer-based models (BERT and CodeBERT) are retrained on prepared sets of labeled URLs and demonstrate high performance and classification accuracy, but do not provide an explicit explanation for the decision made to classify URLs.

Domain-specific models typically achieve over 95 % F1-score on data with known distribution, but have low generalization ability. Achieving high performance on test data from the same set as the training set is not a unique phenomenon. However, the application of models to new data from other sources often shows a significant decrease in classification quality (an increase in type 1 and type 2 errors). The work [16] shows that URLNet and URLTran models trained on the same dataset lose 10–30 % of the F1-score on URLs from other sources, which is due to “data drift”. Therefore, it is necessary to combine approaches based on transfer learning (additional training of specialized high-speed models) on new data with models that are capable of explaining the classification results, providing significant support for the expert making the final decision.

**Development of the phishing link analysis system with mechanisms for explaining decisions taken.** The structural diagram of the phishing link analysis system includes modules that allow not only to classify a URL link as phishing or safe, but also to provide detailed explanations for the expert (Figure 1).

[2] protectai. llm-guard: The security Toolkit for LLM Interactions. GitHub. URL: https://github.com/protectai/llm-guard/tree/main [Accessed 29th August 2025].

Module (1) for preparing initial data and enriching context performs the extraction of lexical and statistical features, and also checks whether the URL domain belongs to the lists of known domains, collects data on the age of registration (WHOIS), etc. More than 89 features are generated, including comprehensive information about the URL.

The module (2) for creating and managing data for training models allows to replenish the database (DB1) with new labeled data from external sources (for example, PhishTank subscription bulletins) and internal data validated by specialists from the SOC. A constantly updated example database allows for additional training and/or retraining of classifier models as data "drifts". This module also collects a database of explanations of signs that a URL belongs to phishing links (for example, in the form of a comparison with MITRE tactics and techniques, known APT attacks, etc.).

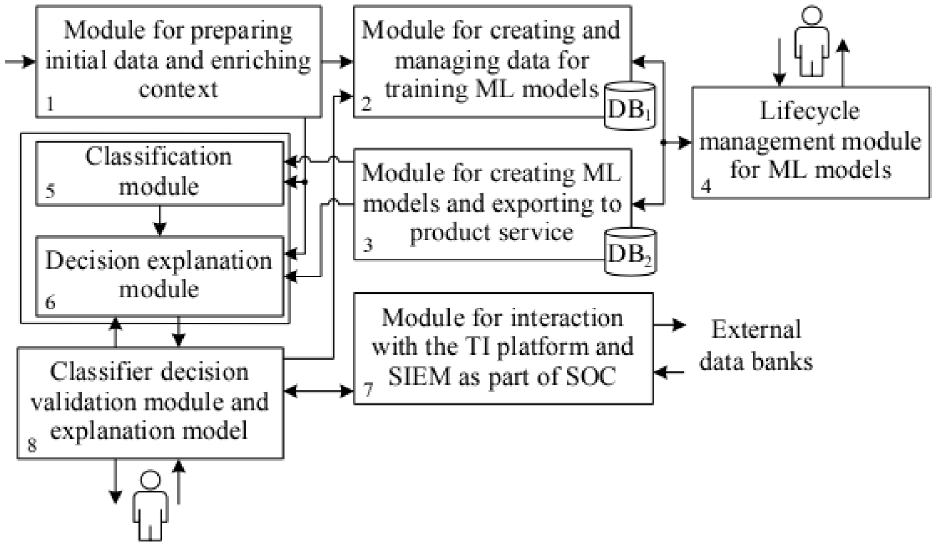


Figure 1 – Structural diagram of the phishing link analysis system
Рисунок 1 – Структурная схема системы анализа фишинговых ссылок

Module (3) for creating ML models and exporting them to a product service allows you to update models that operate as part of the data processing pipeline.

The data analyst uses module (4) to manage the life cycle of models.

Classification module (5) includes models based on committees of decision trees and transformer models that solve the problems of URL link classification.

Module (6) for explaining the decision being made includes interaction via API with a locally deployed LLM – QwQ-32B is the quantitated reasoning model of the Qwen series. The data prepared in module (2) with examples of explanations of the URL belonging to phishing links are used as Zero shot learning in the form of a specially developed prompt. The result of the module (6) is a JSON file with a class label for the URL and a short text explanation of the classification result according to the proposed template. Only those URLs that are classified as malicious by the main model pass through the explanation mechanisms, which allows achieving acceptable overall system performance, due to the significantly (by two orders of magnitude) lower performance of LLM in terms of the number of events per unit of time.

The module (8) for validating classifier decisions and explanation models is an interface for first- and second-line SOC monitoring experts, allowing them to correct the labeling in Reinforcement learning mode.

The module (7) for interaction with the Thread Intelligence platform and SIEM as part of the SOC allows collecting information about the URL context to expand the capabilities of classifiers and explanation modules.

A structural and functional organization and prototype of a system for detecting phishing links based on software and hardware implementation of machine learning methods have been developed (Figure 2).

The proposed algorithms and model for analyzing phishing links are implemented as software in Python with an assessment of effectiveness on real data.

The project consists of a group of containers, each of which performs a separate task, and their interaction provides full coverage of the process of analysis, verification and notification of potential threats:

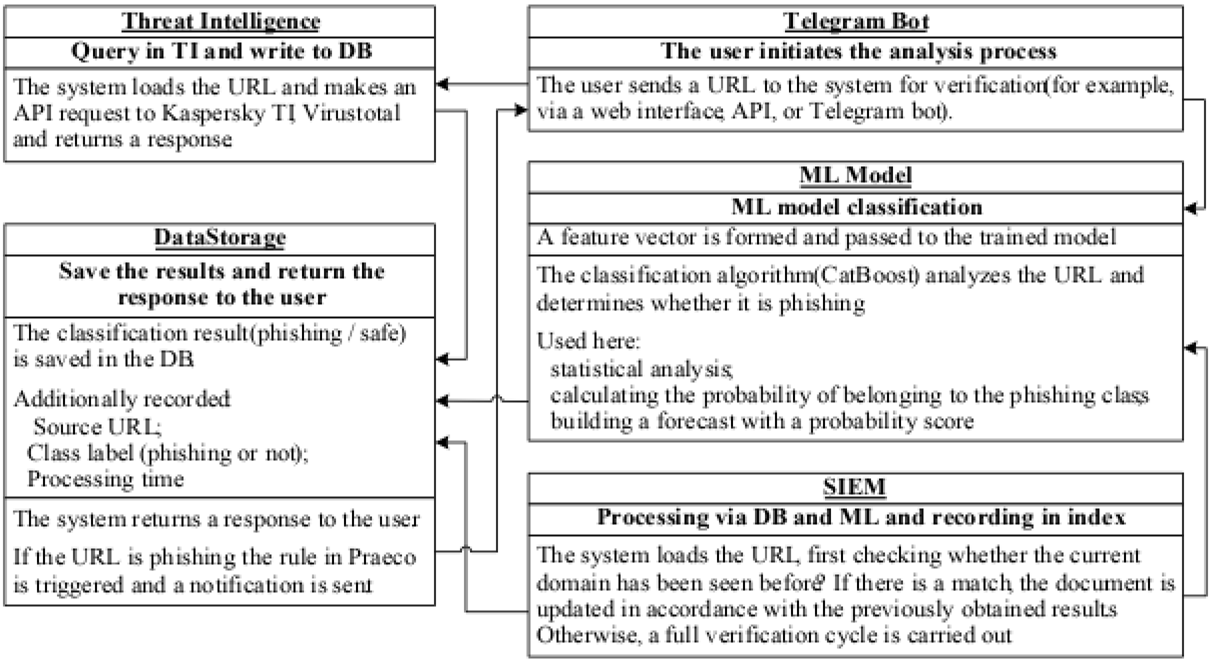


Figure 2 – Structural diagram of interaction of modules
Рисунок 2 – Структурная схема взаимодействия модулей

1. “Telegram bot” (App-tg_bot) implements the functionality of receiving incoming messages from users and serves as the first link in the value verification chain.

2. “Machine learning module” – a container with a machine learning model is deployed on the basis of the “ml_service” image, where the “/check_domain” endpoint is implemented via the Flask API. When a POST request with a domain is received, multi-stage processing occurs.

3. “The Threat Intelligence module” (App-ti_service) is a container for integration with external reputation services such as Kaspersky TI, VirusTotal TI and others via an API implemented on FastAPI.

4. “Data storage service” – a database for long-term storage of all system operation results; a container with a MongoDB [17] database is used.

5. Container for updating Elasticsearch (“App-update_es”). In the test environment, the project uses Elasticsearch to analyze domains, where the “update_es” container is responsible for regularly updating the “idecoutm” index. Each time the script is run, a check is made to see if the current domain has been processed before: if a match is found, the document is updated to reflect the new results, and if not, a full check cycle is started. The update occurs every 3 seconds, which ensures that the data in Elasticsearch is up-to-date and allows to quickly track changes in the analyzed domains. With the help of Praeco, a correlation rule was created to

notify analysts of the SOC, which allows for a prompt response to information security incidents.

6. The container management container (“Portainer”), which uses the “portainer/portainer-ce:latest” image, provides a web interface for managing all of the project's Docker containers.

**Computational experiment.** The key stage in developing the automatic phishing link detection system was the formation of an up-to-date data set. Datasets widely used in research (ISCX-2016 Dataset, EBBU-2017 Dataset, HISPAR-Phishstats Dataset) quickly become outdated. At the same time, the quality of the classification model, its ability to identify new attack patterns, sensitivity and specificity directly depend on the correctness and relevance of the data.

Data was collected from several authoritative sources:

– phishing URLs were obtained from open repositories: Phishing Site URLs (Kaggle), OpenPhish, URLHaus and PhishTank, where malicious link databases are updated daily;

– legitimate URLs were selected from Alexa and Majestic Million Top 1 Million Websites lists, as well as downloaded from secure corporate proxy logs.

The initial sample contained over 4 million rows, of which, after cleaning and filtering, approximately 3.2 million unique URLs remained, balanced across classes. The cleanup included removal of:

– outdated and “dead” links that do not respond to HTTP requests;

– duplicates;

– informal or too short notes;

– links containing non-domain parameters (e.g. BASE64-encoded parts passed after # or?).

URL normalization was performed in several stages:

– removal of prefixes (http://, https://, www.);

– selecting only the domain name and main subdomains;

– lowercase conversion;

– replacing Unicode characters with ASCII (punycode) to combat homographs.

Data labeling was performed semi-automatically. Some URLs were labeled based on the source (for example, if a link is taken from OpenPhish, it is a priori phishing). In complex cases, the following were used:

– cross-checks via VirusTotal API;

– built-in Reputational Score checking module;

– manual check by WHOIS data (registration date, TLD zone, domain activity).

– validation using OpenAI and QwQ-32B models in batch mode.

The formation of features became the next key stage. Each URL was transformed into a fixed-length vector containing the following feature categories:

– structural: line length, number of characters, ratio of letters to numbers, number of dots, level of subdomain nesting;

– lexical: presence of suspicious words (login, verify, update, secure), use of rare combinations of characters, frequency of bigrams;

– statistical: symbolic entropy (according to Shannon), frequency of use of special symbols, diversity index (symbolic uniqueness);

– morphological: TLD type (.xyz, .top, .ru, .com), frequency of such domains in phishing;

– reputational (if available): domain age, reverse link density (according to Majestic), presence in vendor blacklists.

As a result, a data set was formed containing:

– ~1.6 million phishing links;

– ~1.6 million legitimate links.

The following models were used as a binary classifier (Tables 3, 4).

Table 3 – Description of binary classifiers
Таблица 3 – Описание бинарных классификаторов

| Model | Type | Batch performance evaluation (preprocessing and classification) | Learning mode / inference during integration |
|---|---|---|---|
| Random Forest | Ensembles of decision trees | Less than 5 ms per URL | GPU / CPU |
| XGBoost | | | |
| CatBoost | | | |
| CatBoost C-compiled | | | |
| Code-BERT ONNX | Pre-trained transformer | 80 ms | GPU / GPU |

*First Stage of Testing.* With a similar metric, CatBoost [7] shows the best ROC-AUC metric of 0.924, indicating high ability to discriminate between classes (Table 4). For the best CatBoost model, the False Positive Rate metric was 19.98 %, and the False Negative Rate metric was 8.66 %. With comparable performance of models based on ensembles of decision trees, the CatBoost model was selected, which has the best F1-score indicators on the test sample, but the preparation of the data preprocessing pipeline and the speed of model training are significantly higher. The model also allows to evaluate the significance of each feature during classification. The CatBoost model provides the best representation of categorical and quantitative features with the ability to be efficiently multi-threaded on CPU. When translating the model into the C language and then compiling it using LLVM, the model's performance increased by 26 % compared to the original version.

Additionally, a classifier was built based on the retrained Code-BERT model, deployed in the ONNX model format in the FastAPI container. The ONNX model format is a static computational graph in which the vertices are computational operators, and the edges are responsible for the sequence of data transfer across the vertices, which allows the model to run 1.5–2 times faster in classification mode. But using the model on a server with a CPU in multi-threaded mode loses to models based on CatBoost.

Table 4 – Comparative results of the models on the prepared data set
Таблица 4 – Сравнительные результаты работы моделей на подготовленном наборе данных

| Metrics | CatBoost | RandomForest | XGBoost | CodeBERT Fine-tuned |
|---|---|---|---|---|
| ROC-AUC | 0.924 | 0.915 | 0.918 | 0.967 |
| Accuracy | 0.910 | 0.890 | 0.900 | 0.970 |
| Recall | 0.880 | 0.850 | 0.860 | 0.963 |
| F1-score | 0.900 | 0.870 | 0.880 | 0.950 |

The peak performance estimate for the number of requests processed in the Security Operation Center (SOC) was: 231 thousand per day, 11 thousand per hour, 100 per second.

*Second Stage of Testing.* The second phase of the evaluation involved deploying the complete system on a test bed simulating the conditions of the SOC. The flow of incoming URLs was formed from several sources:

– corporate mail gateway logs;
– traffic through proxy servers and web filters;
– specially generated requests through a Telegram bot.

The total volume of the test sample was about 100 thousand URLs, arriving at an average speed of 500 requests per minute. Under this load, the system demonstrated stable operation, with an average processing time of 47 ms for one request, including preprocessing, calling the model, and writing the results to the database. The peak load was up to 1100 URLs/minute, while no timeout errors or microservice freezes were recorded.

*Third Stage of Testing.* During integration with Praeco, an attack was simulated: within 5 minutes, the system received 20 fake URL links using the secure-update-login[.]com subdomain and replacing Latin characters with Cyrillic. The system worked with high accuracy, generated a trigger based on the correlation rule and sent a notification to the SOC Telegram channel (Figure 3).

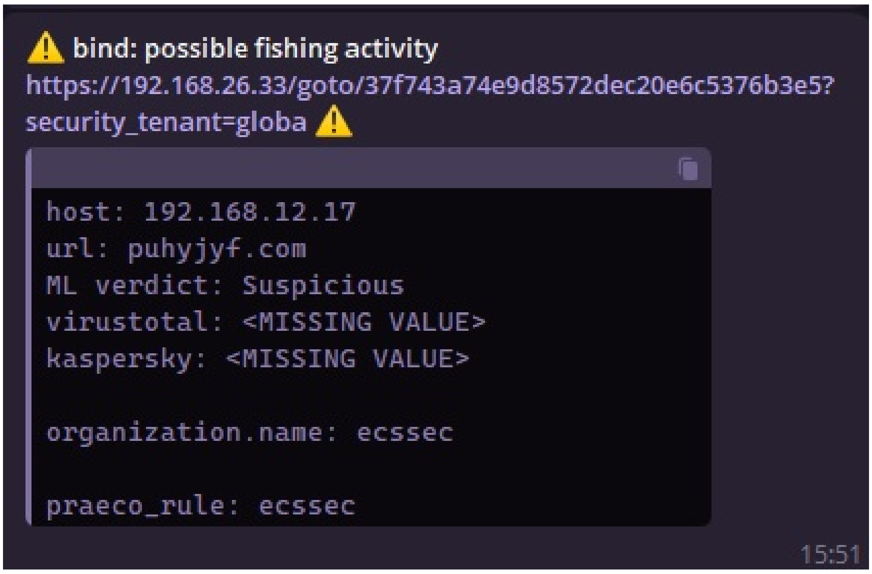


Figure 3 – ML trigger notification
Рисунок 3 – Оповещение о сработке ML

*The Fourth Stage of Testing.* In addition, manual testing was conducted on "zero day" links – URLs that were not previously in the database and which imitated new phishing patterns. Of the 100 such links, the system identified 92 as suspicious, including 6 that had no signs in the VirusTotal databases, but had an anomalous structure.

For an objective assessment, the system was compared with open tools:

– Google Safe Browsing;
– OpenPhish Detection API;
– URLhaus API.

The proposed model detected 15–20 % more "fresh" phishing links, especially in cases of obfuscated addresses, short redirects, and character substitution. A significant advantage was provided by the explainability of classifications – the SOC analyst could immediately see on the basis of which features the link was classified as phishing.

*Testing the Subsystem for Explaining Positive Responses.* By analogy with [16], the application of a subsystem for explaining the reasons for classifying links as phishing is shown for the expanded QwQ-32B model and the OpenAI GPT-4o-mini-high model (via API) (Table 5). The latest versions of the models are used.

Table 5 – Results of explaining the assigned URL class label
Таблица 5 – Результаты объяснения классификации ссылки

| **Model** | **LLM Request and Response** |
|---|---|
| | You are an intelligent assistant. For the given URL, return only the specific URL parts that the explanation considers benign or phishing. You must use the format Benign:Phishing and detailed explain your decision http://www.vivscreisveci.vcirveseiaveesi.ghqphy.top/uWBR vZ8quj /page1.php |
| QwQ-32B | Phishing: www.vivscreisveci.vcirveseiaveesi.ghqphy.top/uWBRvZ8quj /page1.php<br>Detailed Explanation:<br>– The protocol scheme itself is a standard component indicating the use of the Hypertext Transfer Protocol. On its own, “http://” is neutral—it merely specifies how the resource should be fetched.<br>– Phishing (www.vivscreisveci.vcirveseiaveesi.ghqphy.top/uWBRvZ8 quj/page1.php).<br>1. Obfuscated domain: The subdomain “vivscreisveci.vcirveseiaveesi” is a long, nonsensical string designed to hide the true destination or imitate legitimate patterns.<br>2. Suspicious TLD: The “.top” top level domain is inexpensive and often abused by attackers for phishing or malware hosting.<br>3. Random path: The URI path “uWBRvZ8quj/page1.php” lacks meaningful keywords and appears autogenerated—common in phishing links used for tracking or redirecting.<br>4. Lack of HTTPS: Absence of TLS encryption (no “https://”) is atypical for legitimate sites, especially those handling sensitive data. |
| Chat-GPT-4o-mini-high | Phishing: vivscreisveci, vcirveseiaveesi, ghqphy.top, uWBR vZ8quj<br>Explanation of decisions:<br>– http (benign).<br>– Using “http” simply denotes the Hypertext Transfer Protocol. While it’s insecure compared to HTTPS, it’s a standard URL component and in itself not malicious.<br>– page1.php (benign).<br>A PHP file named “page1.php” is a very common, generic filename. Attackers often use such innocuous-looking filenames to hide malicious intent, but the name alone is not proof of phishing.<br>– vivscreisveci, vcirveseiaveesi (phishing).<br>These subdomains consist of unreadable, random character strings with no meaningful relation to any legitimate brand or service. Attackers often employ multiple levels of nonsensical subdomains to obfuscate the true destination.<br>– ghqphy.top (phishing).<br>The second-level domain “ghqphy” combined with the uncommon “.top” TLD is highly suspicious. Legitimate organizations rarely register domains under “.top” – it’s frequently used for throwaway or malicious sites.<br>– uWBR vZ8quj (phishing).<br>The random, space-containing path segment “uWBR vZ8quj” is not human-readable. Legitimate sites use clear, descriptive paths. Such gibberish is a red flag for automated phishing setups. |

The detailed explanation is used to justify a positive classification for SOC first-line monitors and allows them to agree or mark the alarm as a false positive for further adjustment of the training data. The application of approaches for estimating the SHAP and LIME coefficients at this stage was carried out only at the stage of preparing data for training models, since the obtained results require additional interpretation for monitoring specialists.

Thus, the conducted testing confirmed not only the high accuracy of the model, but also its suitability for deployment in real time. The system is capable of analyzing a large flow of incoming URLs, is scalable, and provides integration with response infrastructure, making it applicable to monitoring centers, highly secure organizations, and even national CERT structures.

## Analysis of results and conclusions

The conducted analysis and computational experiment demonstrate that the developed system with a microservice architecture, combining a Telegram bot for primary request processing, a normalization and feature extraction module, a machine learning model based on CatBoost, as well as individual Threat Intelligence modules, successfully solves the problem of detecting phishing URLs. The results for key metrics confirm the high accuracy of the detection model.

Using transformer models allows to increase the F1-measure by 5 % compared to the CatBoost model, but the URL processing time even in batch mode drops by more than 15 times. When a certain threshold value of events per unit of time is reached, this becomes unacceptable for real-time data analysis in SOC. Using local LLMs with prompt and Zero shot learning formation allows to supplement the results of malicious link classification with a text explanation. This increases the awareness of the first- and second-line SOC monitoring specialists during both routine monitoring and incident investigation mode.

The scientific novelty of the proposed solution lies in the development of a set of models for analyzing a symbolic domain name, based on the construction of an ensemble of classifiers that are optimized for hardware platforms, which allows for increased efficiency of analysis when integrated into existing monitoring systems.

The application of well-known models for interpreting the SHAP and LIME results is the next step of the work: it is proposed to use the obtained coefficients:

– as an additional source of features for binary classifiers;

– to retrain a local LLM of smaller size (up to 3B parameters compared to the current 32B inference mode) to create a rich explanation of the classification results with relatively high performance.

## REFERENCES / СПИСОК ИСТОЧНИКОВ

1. Karpova N.E., Voskanyan I.I. Threat of Social Engineering and Phishing in Modern Information Security. *Digital Technology Security*. 2024;(2):69–78. (In Russ.). https://doi.org/10.17212/2782-2230-2024-2-69-78
   Карпова Н.Е., Восканян И.И. Угроза социальной инженерии и фишинга в современной информационной безопасности. *Безопасность цифровых технологий.* 2024;(2):69–78. https://doi.org/10.17212/2782-2230-2024-2-69-78
2. Vasilyev V., Vulfin A., Kuchkarova N. Automation of Software Vulnerabilities Analysis on the Basis of Text Mining Technology. *Voprosy kiberbezopasnosti*. 2020;(4):22–31. (In Russ.).
   Васильев В.И., Вульфин А.М., Кучкарова Н.В. Автоматизация анализа уязвимостей программного обеспечения на основе технологии Text Mining. *Вопросы кибербезопасности*. 2020;(4):22–31.
3. Кутлыев Д.З., Шманина А.В. Использование алгоритмов машинного обучения для защиты от URL-фишинга. В сборнике: *Мавлютовские чтения: Материалы XV Всероссийской молодежной научной конференции*: *Том 4, 26–28 октября 2021 года, Уфа, Россия.* Уфа: Уфимский государственный авиационный технический университет; 2021. С. 430–435.
4. Tonkal Ö., Polat H., Başaran E., Cömert Z., Kocaoğlu R. Machine Learning Approach Equipped with Neighbourhood Component Analysis for DDoS Attack Detection in Software-Defined Networking. *Electronics*. 2021;10(11). https://doi.org/10.3390/electronics10111227
5. Alshingiti Z., Alaqel R., Al-Muhtadi J., Haq Q.E.U., Saleem K., Faheem M.H. A Deep Learning-Based Phishing Detection System Using CNN, LSTM, and LSTM-CNN. *Electronics*. 2023;12(1). https://doi.org/10.3390/electronics12010232

6. Karim A., Shahroz M., Mustofa Kh., Belhaouari S.B., Joga S.R.K. Phishing Detection System Through Hybrid Machine Learning Based on URL. *IEEE Access*. 2023;11:36805–36822. https://doi.org/10.1109/ACCESS.2023.3252366
7. Vasilyev V., Vulfin A., Kuchkarova N. Assessment of Current Threats to Information Security Using Transformer Technology. *Voprosy kiberbezopasnosti*. 2022;(2):27–38. (In Russ.).
Васильев В.И., Вульфин А.М., Кучкарова Н.В. Оценка актуальных угроз безопасности информации с помощью технологии трансформеров. *Вопросы кибербезопасности*. 2022;(2):27–38.
8. Lukmanova K.A., Kartak V.M. Recognition of Phishing Links Using Machine Learning Methods. *Digital Technology Security*. 2024;(3):9–20. (In Russ.). https://doi.org/10.17212/2782-2230-2024-3-9-20
Лукманова К.А., Картак В.М. Распознавание фишинговых ссылок с использованием методов машинного обучения. *Безопасность цифровых технологий*. 2024;(3):9–20. https://doi.org/10.17212/2782-2230-2024-3-9-20
9. Arrieta A.B., Díaz-Rodríguez N., Del Ser J., et al. Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges Toward Responsible AI. *Information Fusion*. 2020;58:82–115. https://doi.org/10.1016/j.inffus.2019.12.012
10. Mahdaouy A.E., Lamsiyah S., Idrissi M.J., Alami H., Yartaoui Z., Berrada I. DomURLs_BERT: Pre-Trained BERT-Based Model for Malicious Domains and URLs Detection and Classification. arXiv. URL: https://arxiv.org/abs/2409.09143 [Accessed 29th August 2025].
11. Maneriker P., Stokes J.W., Lazo E.G., Carutasu D., Tajaddodianfar F., Gururajan A. URLTran: Improving Phishing URL Detection Using Transformers. In: *MILCOM 2021 – 2021 IEEE Military Communications Conference (MILCOM), 29 November – 02 December 2021, San Diego, CA, USA*. IEEE; 2021. P. 197–204. https://doi.org/10.1109/MILCOM52596.2021.9653028
12. Yang Yu, Li H., Jing D. Detection of Malicious URL Based on BERT-CNN. In: *2023 International Conference on Computer Science and Automation Technology (CSAT), 06–08 October 2023*, *Shanghai, China*. IEEE; 2023. P. 284–288. https://doi.org/10.1109/CSAT61646.2023.00079
13. Tsai Yu.-D., Liow C., Siang Y.-Sh., Lin Sh.-D. Toward More Generalized Malicious URL Detection Models. In: *AAAI 2024: Thirty-Eighth AAAI Conference on Artificial Intelligence, IAAI 2024: Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, EAAI 2014: Fourteenth Symposium on Educational Advances in Artificial Intelligence, 20–27 February 2024, Vancouver, Canada*. AAAI Press; 2024. P. 21628–21636. https://doi.org/10.1609/aaai.v38i19.30161
14. Rao R.S., Vaishnavi T., Pais A.R. CatchPhish: Detection of Phishing Websites by Inspecting URLs. *Journal of Ambient Intelligence and Humanized Computing*. 2020;11(2):813–825. https://doi.org/10.1007/s12652-019-01311-4
15. Alsowail R.A. Anomaly Detection Based Capsnet for Malicious URL Detection System. *Wireless Networks*. 2025;31:3785–3801. https://doi.org/10.1007/s11276-025-03960-0
16. Rashid F., Ranaweera N., Doyle B., Seneviratne S. LLMs Are One-Shot URL Classifiers and Explainers. *Computer Networks*. 2025;258. https://doi.org/10.1016/j.comnet.2024.111004
17. Mailewa A., Mengel S., Gittner L., Khan H. Mechanisms and Techniques to Enhance the Security of Big Data Analytic Framework with MongoDB and Linux Containers. *Array*. 2022;15. https://doi.org/10.1016/j.array.2022.100236

## ИНФОРМАЦИЯ ОБ АВТОРАХ / INFORMATION ABOUT THE AUTHORS

**Шаймарданов Артур Филюсович,** магистрант Уфимского университета науки и технологий, Уфа, Российская Федерация.
*e-mail:* shaimardanov.af@ugatu.su

**Arthur F. Shaimardanov,** Master's Degree student, Ufa University of Science and Technology, Ufa, the Russian Federation.

**Вульфин Алексей Михайлович,** доктор технических наук, профессор Уфимского университета науки и технологий, Омского государственного технического университета, Омск, Российская Федерация.
*e-mail:* vulfin.alexey@gmail.com
ORCID: 0000-0001-5857-2413

**Alexey M. Vulfin,** Doctor of Engineering Sciences, Professor, Ufa University of Science and Technology, Omsk State Technical University, Omsk, the Russian Federation.

**Кириллова Анастасия Дмитриевна,** кандидат технических наук, доцент Уфимского университета науки и технологий, Уфа, Российская Федерация.
*e-mail:* kirillova.andm@gmail.com
ORCID: 0009-0000-4164-2526

**Anastasia D. Kirillova,** Candidate of Engineering Sciences, Associate Professor, Ufa University of Science and Technology, Ufa, the Russian Federation.

**Минко Александр Васильевич,** аспирант Уфимского университета науки и технологий, Уфа, Российская Федерация.
*e-mail:* alex_shtem@mail.ru

**Alexander V. Minko,** Postgraduate, Ufa University of Science and Technology, Ufa, the Russian Federation.

*Статья поступила в редакцию 03.09.2025; одобрена после рецензирования 15.10.2025; принята к публикации 27.10.2025.*

*The article was submitted 03.09.2025; approved after reviewing 15.10.2025; accepted for publication 27.10.2025.*