http://doi.org/10.35784/iapgos.6366 *received: 03.07.2024 | revised: 23.10.2024 | accepted: 20.11.2024 | available online: 21.12.2024*

USING SUPPORT VECTORS TO BUILD A RULE-BASED SYSTEM FOR DETECTING MALICIOUS PROCESSES IN AN ORGANISATION'S NETWORK TRAFFIC

Halyna Haidur, Sergii Gakhov, Dmytro Hamza
State University of Information and Communication Technologies, Department of Information and Cyber Security, Kyiv, Ukraine

Abstract. The growing complexity and sophistication of cyberattacks on organisational information resources and the variety of malware processes in unprotected networks necessitate the development of advanced methods for detecting malicious processes in network traffic. Systems for detecting malicious processes based on machine learning and rule-based methods have their advantages and disadvantages. We have investigated the possibility of using support vectors to create a rule-based system for detecting malicious processes in an organisation's network traffic. We propose a method for building a rule-based system for detecting malicious processes in an organisation's network traffic using the distribution data of the relevant features of support vectors. The application of this method on real CSE-CIC-IDS2018 network traffic data containing characteristics of malicious processes has shown acceptable accuracy, high clarity and computational efficiency in detecting malicious processes in network traffic. In our opinion, the results of this study will be useful in creating automatic systems for detecting malicious processes in the network traffic of organisations and in creating and using synthetic data in such systems.

Keywords: network security, classification of network traffic, supervised learning, support vector machine classification, rule-based systems

WYKORZYSTANIE WEKTORÓW WSPIERAJĄCYCH DO ZBUDOWANIA OPARTEGO NA REGUŁACH SYSTEMU WYKRYWANIA ZŁOŚLIWYCH PROCESÓW W RUCHU SIECIOWYM ORGANIZACJI

Streszczenie. *Rosnąca złożoność i wyrafinowanie cyberataków na zasoby informacyjne organizacji oraz różnorodność procesów złośliwego oprogramowania w niezabezpieczonych sieciach wymagają opracowania zaawansowanych metod wykrywania złośliwych procesów w ruchu sieciowym. Systemy wykrywania złośliwego oprogramowania oparte na uczeniu maszynowym i metodach regułowych mają swoje zalety i wady. Zbadaliśmy możliwość wykorzystania wektorów wspierających do stworzenia opartego na regułach systemu wykrywania złośliwych procesów w ruchu sieciowym organizacji. Proponujemy metodę budowania hybrydowego systemu regułowego do wykrywania złośliwych procesów w ruchu sieciowym organizacji przy użyciu danych o dystrybucji odpowiednich cech wektorów podporowych. Zastosowanie tej metody na rzeczywistych danych o ruchu sieciowym CSE-CIC-IDS2018 zawierających charakterystykę złośliwych procesów wykazało akceptowalną dokładność, wysoką zrozumiałość i wydajność obliczeniową w wykrywaniu złośliwych procesów w ruchu sieciowym. Naszym zdaniem wyniki tego badania będą przydatne w tworzeniu automatycznych systemów wykrywania złośliwych procesów w ruchu sieciowym organizacji oraz w tworzeniu i wykorzystywaniu danych syntetycznych w takich systemach.*

Słowa kluczowe: bezpieczeństwo sieci, klasyfikacja ruchu sieciowego, uczenie nadzorowane, klasyfikacja maszyn wektorów nośnych, systemy oparte na regułach

Introduction

The presence of malicious processes in network traffic is a sign of a breach of an organisation's information system. Malicious processes occur during the operation and interaction of malicious software, as well as a result of various types of attacks. One of the ways to ensure cybersecurity of organisations' information systems is to monitor network traffic, detect malicious processes and respond to them accordingly. Network traffic monitoring also allows specialists to understand the data flow, communication patterns, and potential vulnerabilities in organisations' information systems.

Advanced network traffic monitoring solutions use behavioural analysis and machine learning algorithms to detect abnormal patterns and behaviour. This helps to detect zero-day attacks and sophisticated threats that are not detected by traditional security methods and tools. Network traffic monitoring tools often integrate with threat intelligence feeds to provide up-to-date information on known threats and compromise indicators. Network traffic monitoring solutions should provide continuous monitoring of network activity and generate real-time alerts for suspicious or malicious events. This will allow security teams to respond quickly to potential threats.

It is logical that the systems for detecting malicious processes in the network traffic of organisations are subject to requirements for classification accuracy and computing resource consumption. When creating and applying systems for detecting malicious processes in the network traffic of organisations based on machine learning methods, additional requirements should be put forward to ensure trust in these systems and confidence in their performance [14]. Therefore, in our study, we also focus on the requirements for explainability and interpretability of the malware detection system and its performance results.

Indeed, to detect malicious processes in the network traffic of organisations, detection systems based on machine learning methods, in particular, support vector machine (SVM), and rule-based systems can be used, each of which has its advantages and disadvantages.

SVM is effective in high-dimensional spaces, which makes it suitable for detecting malicious processes in the network traffic of organisations where features can span dozens or even hundreds of dimensions. SVM has high generalisation capabilities, which allows it to process implicit, complex data and adapt to new or unknown classes of malicious processes. SVM naturally solves binary classification problems, making it suitable for detecting malicious network traffic.

At the same time, SVM are often considered "black box" models, as the decision-making process in them is not easy to interpret for humans. It can be difficult to understand why a particular solution was obtained. When applying SVM, there are also difficulties in interpreting and understanding the importance of specific features in detecting malicious processes in organisations' network traffic. SVM are sensitive to the choice of kernel function and hyperparameter settings that affect their performance.

In [9], it is noted that a rule-based system refers to a knowledge-based system, where the knowledge base is represented in the form of rules or their sets (rule systems). Rules are a clear, simple and flexible means of expressing knowledge.

Rule-based systems are easy to interpret, as they operate on the basis of clear rules or their sets (rule systems) that are understandable to humans. Rules or their sets (rule systems) reflect both domain-specific knowledge and subject matter expertise, making them flexible and adaptive to the specific characteristics of malicious processes in organisations' network traffic. Rule-based systems are relatively easy to build.

Adding, modifying, or deleting rules and rule sets (rule systems) is done with minimal effort.

At the same time, rule-based systems show a weak ability to generalise, especially when faced with new or unknown variants of malicious process patterns in an organisation's network traffic that do not match existing rule sets. As the number of rules and their sets (rule systems) grows, managing such a detection system can become a daunting task. Overly complex rule systems and their number can lead to redundancy or conflicts between them. This will lead to a large number of false alarms and, consequently, overload the response teams.

Creating effective rules and their sets (rule systems) requires in-depth knowledge of the specifics of the domain and the subject area as a whole, and requires a lot of manual work. It can be time-consuming and difficult to scale for large data sets or rapid changes in the volume and content of malicious processes in organisations' network traffic.

Consequently, support vector machine-based detection systems will have strong generalisation and performance in high-dimensional spaces but lack interpretability, while rule-based systems provide explainability and interpretability and integrate domain and subject matter expertise, but may have problems with generalisation and scalability. The choice between these approaches often depends on the specific requirements of the organisation's network traffic malware detection system, including the need for interpretability, the complexity of the organisation's network traffic, the available domain expertise, and the desired balance between accuracy and clarity. Combining both approaches in a hybrid system can also leverage their strengths to improve the overall effectiveness of detecting malicious processes in an organisation's network traffic.

In our opinion, it is the hybrid system for detecting malicious processes in the network traffic of organisations using machine learning methods (in particular, SVM) that will allow us to obtain a model, extract "informative" support vectors and, on their basis, form rules and their sets (systems). In doing so, we will gain advantages in the practical application of such a hybrid detection system.

Our research is fundamentally different from others in the following respects:

- we have thoroughly investigated the potential advantages and disadvantages of detection systems based on SVM and rule-based systems and proposed a methodology for creating a hybrid system for detecting malicious processes in an organisation's network traffic that will meet the requirements of acceptable detection (classification) accuracy, minimal consumption of computing resources, high explainability and interpretability, which is important for the implementation of automatic detection systems;
- we have selected and used real data of CSE-CIC-IDS2018 network traffic, which is an important point in terms of further creation and use of synthetic data in automatic systems for detecting malicious processes in the network traffic of an organisation;
- the nature of the input data led us to choose a powerful machine learning algorithm based on SVM for the study;
- testing of the proposed methodology for creating a hybrid system for detecting malicious processes in the network traffic of an organisation was carried out on the CSE-CIC-IDS2018 dataset and sufficiently high estimates of malicious process detection were obtained (*precision = 0.74, recall = 1.00, F1-score = 0.85*).

The significance of our research is that by combining the power of the support vector method with the ability to interpret the results of a rule-based system, we can achieve a balance between accuracy and human understanding, which can ultimately lead to more effective and informed practices for detecting malicious processes in an organisation's network traffic.

The rest of the paper is structured as follows: Section 1 is a review of works on the use of support vectors to create rules and their sets (rule systems). Section 2 contains the results of the analysis of the initial data set and reveals the proposed research methodology. Section 3 presents the results of the analysis of the obtained results. In Section 4, we discuss the results obtained. Section 5 contains the conclusions, and Section 6 contains directions for future research.

1. Related works

In [5], it is noted that rule extraction belongs to the group of post-hoc methods of Explainable Artificial Intelligence (XAI). This group of methods is applied to an already trained machine learning model (usually a black box) to explain the forecasting results obtained with its help.

Rule extraction methods are differentiated into two subgroups: model-specific and model-agnostic [2]. Model-specific methods allow to generate rules based on specific information from the trained model, while model-agnostic methods use only input and output information from the trained model, hence they can be applied to any other model.

Post-hoc XAI methods are mainly differentiated according to whether they provide local explanations (explanations for a specific data point) or global explanations (explanations for the entire model). The advantage of most rule extraction methods is that they provide explanations for both cases simultaneously [2].

The authors of [3] note that the task of extracting rules from SVM is to create rules from the SVM model, not directly from the data. Thus, the explanation of the patterns learned and embedded in their structure (the model of support vectors and associated parameters) is revealed and provided to end users in an understandable form.

In [8], the authors proposed a method for extracting reduced rules based on biased random forest and fuzzy support vector machine is proposed. Biased random forest uses the k-nearest neighbours (k-NN) algorithm to identify critical samples and generates more trees that tend to diagnose diabetes based on critical samples to improve the tendency of the generated rules for diabetic patients. In addition, the conditions and rules are reduced based on the error rate and coverage rate to enhance interpretability.

In [12], the authors proposed a rule extraction method to identify the impact of investor, stock and corporate performance characteristics on stock trading preferences. The authors used a hybrid system to combine the advantages of two approaches: SVM as an accurate classifier and decision tree (DT) as a generator of interpreted models.

In [10], the authors combined linear regression and SVM classifiers in decision trees. These methods are used to extract heuristic rules from datasets one by one. The authors propose to use linear classifiers, which, in our opinion, are not entirely effective methods for datasets with nonlinear relationships.

In [20], to improve the comprehensibility of SVM, the authors proposed a technique for extracting rules from SVM by analysing the distribution of samples. To do this, a consistent sample region is defined in terms of the classification boundary and a consistent coverage of the sample space is formed. Then, a coverage reduction algorithm is applied to extract a compact representation of the classes, thus deriving a minimal set of decision rules.

In [6], it is noted that SVM are "the most advanced data mining tool". The strength of SVM is also their main drawback, as "the generated nonlinear models are usually considered to be incomprehensible black box models". This problem, especially for critical industries, is solved by extracting rules from SVM models to mimic their behaviour and make them understandable [6, 18, 19].

Paper [7] describes an algorithm for converting linear SVM and any other arbitrary linear classifiers based on hyperplanes into a set of non-overlapping rules that, unlike the original classifier, can be easily interpreted by humans. Each iteration of the rule extraction algorithm is formulated as a constrained optimisation problem that does not require large computational costs.

In [15, 16], the authors proposed a rule extraction method, the essence of which is to determine the support vectors, vertex points, and prototype points for each class using a clustering algorithm. The identified points are used to obtain a geometric surface that encloses the remaining data points inside. There are two ways to do this. The first way is to draw hyperrectangles to define interval rules. The second way is to draw ellipsoids to define equation rules.

In our opinion, the construction of hyperrectangles and ellipsoids for the purpose of further defining rules will lead to less accuracy of the created rule-based system both because of incomplete coverage of space and because there is no need to close the space with geometric surfaces.

Paper [4] proposes a method for extracting rules from support vectors using a modified sequential coverage algorithm. The rules are generated based on an ordered search for the most discriminative features measured by the interclass separation. The effectiveness of the rules is evaluated using the measured true and false alarm rates and the Compute Area Under the Receiver Operating Characteristic Curve (ROC AUC).

In our opinion, relying on the number and nature of the distribution of accurately classified positive and negative support vector examples when applying powerful SVM kernel functions is not entirely effective.

In [13], the authors proposed an approach to extract rules from a trained SVM model by explicitly using support vectors and the observation that they are usually close to the decision boundary. Active learning involves focusing on obvious problem areas, which for rule induction methods are areas close to the decision boundary of the SVM, where most of the noise is located. By generating additional data close to these support vectors, provided by the class label of the trained SVM model, rule induction methods are better at identifying relevant discrimination rules.

In [5], the authors used rule extraction techniques in OneClass SVM models to detect anomalies by generating hypercubes that encapsulate non-anomalous data points and using their vertices as rules to explain when a data point is considered non-anomalous.

In our opinion, hypercubes are not flexible in multi-dimensional spaces. Therefore, we proposed to extract rules from the distribution of feature values by projections of support vectors on the coordinate axes of features that are informative.

2. Method of building a rule-based system using support vectors

2.1. Important theoretical points of the SVM with a RBF kernel

Let's consider some important points of SVM that are relevant to our task.

When applying SVM, the optimal hyperplane is determined to maximise the generalisation ability. However, if the training data is not linearly resolved, the resulting classifier may not have a high generalisation capability, even though the hyperplane is optimally determined. Therefore, to increase the linear resolution, the initial input space is decomposed into a high-dimensional scalar product space called the feature space [17].

A distinctive feature of SVM is that we can select and apply different kernel functions to improve generalisation performance. It is noted in [17] that the choice of kernels for specific applications is very important, and the development of new kernels is an ongoing research topic.

The input data for the classification problem are two finite subsets of vectors $\mathbf{x}$ from the training set

$$(y_1, \mathbf{x}_1),\ \dots ,(y_l, \mathbf{x}_l),\ \mathbf{x} \in \mathrm{R}^n,\ y \in \{-1,\ 1\}$$

The general expression of a set of SVM decision rules is as follows [17]:

$$D(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \tag{1}$$

where $\{\mathbf{x}_i\}_{i=1}^{n}$ are called support vectors, which are a relatively small set of training data near the separating hyperplane.

SVM allows to find a linear separating hyperplane with a maximum boundary in the space of higher features induced by the kernel function $K(\mathbf{x}_i, \mathbf{x})$. The accuracy of the classification depends on the chosen kernel function.

We use the radial basis function (RBF) of the kernel, which has the following form [17]:

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2) \tag{2}$$

where γ is a hyperparameter to control the radius.

We can rewrite formula (2) as follows:

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x}\|^2) \exp(-\gamma \|\mathbf{x}'\|^2) \exp(2\gamma\ \mathbf{x}^{\mathsf{T}} \mathbf{x}') \tag{3}$$

The multiplier $\exp(2\gamma\ \mathbf{x}^{\mathsf{T}}\mathbf{x}')$ of formula (3) can be represented by a polynomial using the Taylor series expression:

$$\exp(2\gamma\ \mathbf{x}^{\mathsf{T}}\mathbf{x}') = 1 + 2\gamma\ \mathbf{x}^{\mathsf{T}}\mathbf{x}' + 2\gamma^2 \left(\mathbf{x}^{\mathsf{T}}\mathbf{x}'\right)^2 + \frac{(2\gamma)^3}{3!}\left(\mathbf{x}^{\mathsf{T}}\mathbf{x}'\right)^3 + \dots \tag{4}$$

Formula (4) represents functions in the form of an infinite sum of terms calculated from the values of the derivative functions at one point [17]. The physical meaning of formula (4) is that its properties allow us to describe the spaces of existence of feature vectors of the training sample of any dimension and any complexity.

The expression of a set of SVM decision rules with an RBF kernel is as follows:

$$D(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}\|^2) + b \tag{5}$$

It should be noted that the support vectors are the centres of radial basis functions. Since the kernels of radial basis functions use Euclidean distance, they are not robust to outliers [17].

When training a classifier, we usually try to maximise the classification performance for the training data. However, if the classifier is too well suited to the training data, then the classification ability on unknown data, i.e. the ability to generalise, deteriorates. This phenomenon is called overfitting. Therefore, a compromise must be made between generalisation ability and training data suitability. It is worth noting that the recognition accuracy rate assesses the overall performance of a classifier and is used to compare them [17].

It is the properties of the RBF kernel that allow the SVM classifier to adapt to the original training data as much as possible, which will lead to overtraining of the model and, as a result, the ability to generalise deteriorates. However, the benefit of using SVM lies in the identification of support vectors as informative data for the classification task. It should be emphasised that the number of support vectors becomes much smaller, and, accordingly, less computational resources are spent when considering the option of having linearly inseparable data, which is mostly common in practice.

2.2. Description of the method of building a rule-based system using support vectors

Determining the content of the rule system and justifying the values of the variables of these rules is important for building effective intrusion detection systems (their configuration) and ensuring network security, as well as explaining the results of applying machine learning algorithms.

In turn, the success of using synthetic data sets for machine learning to detect malicious processes in an organisation's network traffic depends on the quality and accuracy of the data set. A synthetic dataset must be designed to accurately reflect the distribution of real-world data, and it must be diverse enough to cover a wide range of possible attack scenarios. In addition, the machine learning model must be properly trained and validated to ensure that it can effectively detect malicious activity.

The CSE-CIC-IDS2018 dataset [1] includes network traffic data from both normal network activity and simulated attacks and malicious processes, namely Brute-force, Heartbleed, Botnet, DoS, DDoS, web attacks, and insider attacks. The attack infrastructure includes 50 machines, and the victim organisation has five departments and includes 420 machines and 30 servers. The dataset contains network traffic and system logs of each machine, as well as 80 statistical characteristics (features) extracted from the recorded network traffic using the CICFlowMeter-V3 analyser [1].

Our choice of this particular dataset was influenced by the fact that the researchers built a model for the experiments (Fig. 1), which can be considered as a model of the information system of a modern organisation with the appropriate architecture, asset composition, and information resources of Amazon Web Services (AWS). In our opinion, this is a fundamental point in the development of synthetic datasets, the creation of machine learning models and their application in target systems.

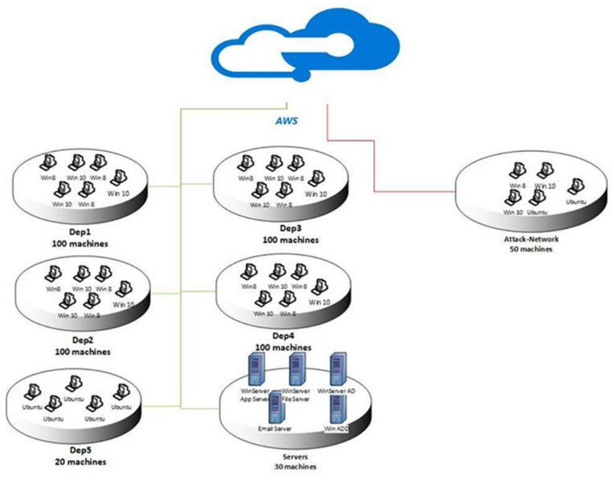


Fig. 1. Model of the information system of a modern organisation [1]

We chose to study a part of the CSE-CIC-IDS2018 *Friday-02-03-2018_TrafficForML_CICFlowMeter.csv* dataset [1], which reflects the operation of botnets such as Zeus and Ares. An important point in this dataset is that only the processes of the botnet functioning were modelled, which are related to the collection and periodic sending of screenshots of the compromised computers (which are part of the botnet) to the control computer. In our opinion, in this case, it becomes more difficult to distinguish between the normal network activity of an organisation's information system and the activity of a botnet.

In [11], a botnet is defined as a network of computers infected with malware. Attackers use botnets consisting of a large number of computers to perform various malicious actions without the users' knowledge.

We believe that a botnet should be viewed as an information system created and used for malicious purposes. Among the many purposes of using a botnet, there are tasks in which its processes will be characterised by low intensity. Therefore, identifying such processes of a botnet against the background of the target information system is a complex task that requires the use of appropriate machine learning methods.

Fig. 2 shows a diagram of the proposed method for building a rule system using support vectors to detect malicious processes in an organisation's network traffic.

During the experiments, we used a computer with the following hardware and software:

- CPU AMD Ryzen 5 5600X 6-Core Processor 3.70 GHz;
- RAM 32 GB;
- SSD Samsung SSD 980 PRO 1TB;
- OS Windows 11 Pro, version 23H2, build 22631.3810.

We used the interactive web-based computing platform Jupyter Notebook as a development environment and libraries of applications for working with data and visualising results, such as pandas, NumPy, scikit-learn, matplotlib, Seaborn, Plotly.

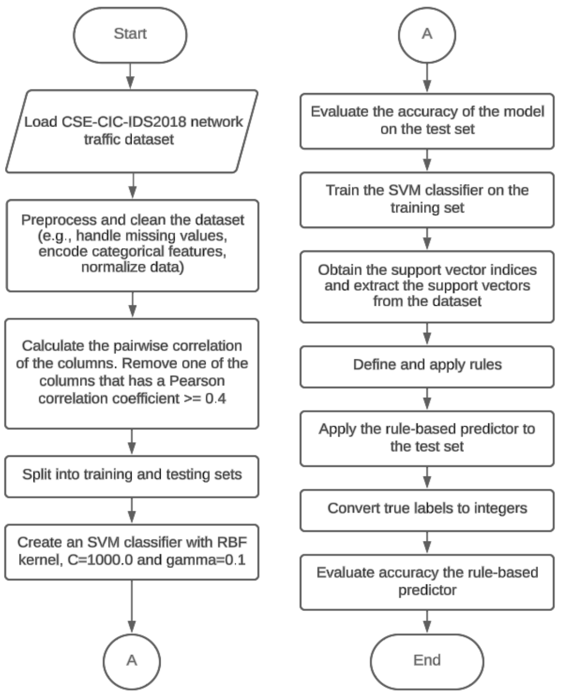

Start
Load CSE-CIC-IDS2018 network traffic dataset
Preprocess and clean the dataset (e.g., handle missing values, encode categorical features, normalize data)
Calculate the pairwise correlation of the columns. Remove one of the columns that has a Pearson correlation coefficient >= 0.4
Split into training and testing sets
Create an SVM classifier with RBF kernel, C=1000.0 and gamma=0.1
A
A
Evaluate the accuracy of the model on the test set
Train the SVM classifier on the training set
Obtain the support vector indices and extract the support vectors from the dataset
Define and apply rules
Apply the rule-based predictor to the test set
Convert true labels to integers
Evaluate accuracy the rule-based predictor
End

Fig. 2. Scheme of the method of building a rule-based system using support vectors to detect malicious processes in the network traffic of an organisation

3. Results and analysis

The preprocessing stage in machine learning involves preparing and converting the source data into a format suitable for efficient training of the machine learning algorithm. The content of the data pre-processing stage of the proposed method (Fig. 2) includes the following steps:

Data Quality Assurance. This step involves processing missing values, outliers, and other inconsistencies in the data.

Feature Encoding. At this stage, categorical variables are converted into a numerical format for processing by an appropriate machine learning algorithm;

Feature Selection/Extraction. In this step, only the most relevant features are selected using correlation values to extract important information from the data.

Feature Scaling. To bring the values of numerical features to a single scale, we used a method to standardise their values;

Handling Imbalanced Data. In our case, the classes in the target variable are unbalanced, so we used a method to perform a random under-sampling.

Splitting Data. In this step, the data is divided into training and test data.

These preprocessing steps are crucial for a machine learning algorithm to effectively learn patterns and relationships from data, leading to better performance and generalisation to unknown data. Let's take a closer look at some of the steps.

We conducted a Pearson correlation analysis and removed highly correlated features (we set the correlation coefficient threshold at 0.4). Pearson's correlation coefficient is calculated to summarise the linear relationship between the values of the features. The Pearson correlation coefficient is calculated by dividing the covariance of two variables by the product of their respective standard deviations. This is the normalisation of the covariance between two variables to produce an interpretable estimate. The use of the mean and standard deviation in the calculation indicates the need for the two data samples to have a Gaussian or similar distribution. The result of the calculation is the corresponding correlation coefficient, which can be used to interpret the relationship. A Pearson correlation coefficient of 0.4 indicates a less significant correlation.

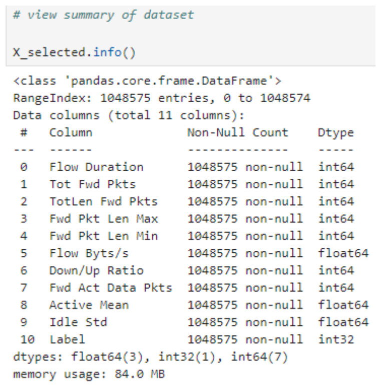
This led to a reduction in the number of features to 10. The features that were selected are shown in Fig. 3.

One way to handle unbalanced datasets is to reduce the number of observations from all classes except the minority class. We used *RandomUnderSampler* as a quick and easy way to balance the data by randomly selecting a subset of the data for the target classes.

In our case, this can be seen as one of the methods to reduce the dimensionality of the dataset under study. The number of rows in the dataset became 572382.

We divided the variables for training and testing with *test_size = 0.3*.

We used the *StandardScaler* tool to standardise the feature values. *StandardScaler* removes the mean and scales the variance to one.

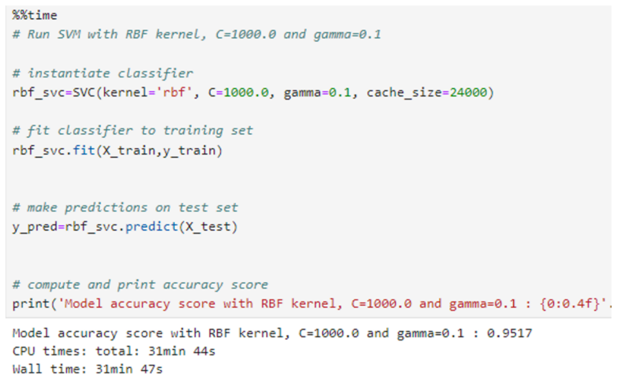


```
# view summary of dataset

X_selected.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1048575 entries, 0 to 1048574
Data columns (total 11 columns):
 #   Column            Non-Null Count    Dtype
---  ------            --------------    -----
 0   Flow Duration     1048575 non-null  int64
 1   Tot Fwd Pkts      1048575 non-null  int64
 2   TotLen Fwd Pkts   1048575 non-null  int64
 3   Fwd Pkt Len Max   1048575 non-null  int64
 4   Fwd Pkt Len Min   1048575 non-null  int64
 5   Flow Byts/s       1048575 non-null  float64
 6   Down/Up Ratio     1048575 non-null  int64
 7   Fwd Act Data Pkts 1048575 non-null  int64
 8   Active Mean       1048575 non-null  float64
 9   Idle Std          1048575 non-null  float64
 10  Label             1048575 non-null  int32
dtypes: float64(3), int32(1), int64(7)
memory usage: 84.0 MB
```

Fig. 3. The list of features obtained

We created a model (classifier) using the SVM machine learning algorithm with the following hyperparameters: RBF kernel, regularisation hyperparameter *C = 1000.0*, and kernel coefficient *gamma = 0.1*. In previous studies, we have determined that these values of the SVM classifier hyperparameters are optimal for the selected data set. The results of applying the SVM algorithm are shown in Fig. 4.

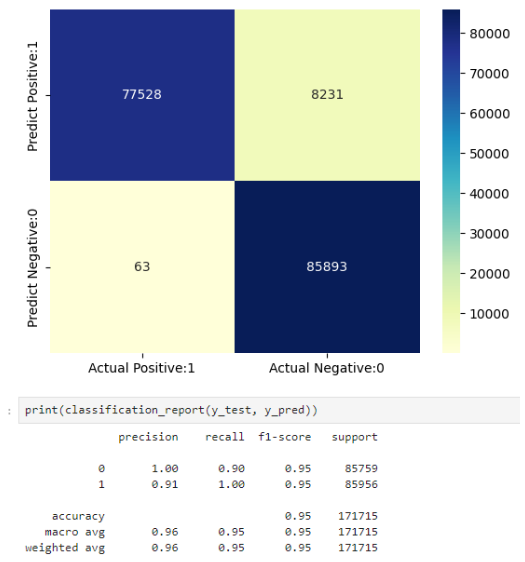


```
%%time
# Run SVM with RBF kernel, C=1000.0 and gamma=0.1

# instantiate classifier
rbf_svc=SVC(kernel='rbf', C=1000.0, gamma=0.1, cache_size=24000)

# fit classifier to training set
rbf_svc.fit(X_train,y_train)


# make predictions on test set
y_pred=rbf_svc.predict(X_test)


# compute and print accuracy score
print('Model accuracy score with RBF kernel, C=1000.0 and gamma=0.1 : {0:0.4f}'.
```

```
Model accuracy score with RBF kernel, C=1000.0 and gamma=0.1 : 0.9517
CPU times: total: 31min 44s
Wall time: 31min 47s
```

Fig. 4. Results of applying the SVM algorithm

The values of the estimates of the results of applying the SVM algorithm are shown in Fig. 5.

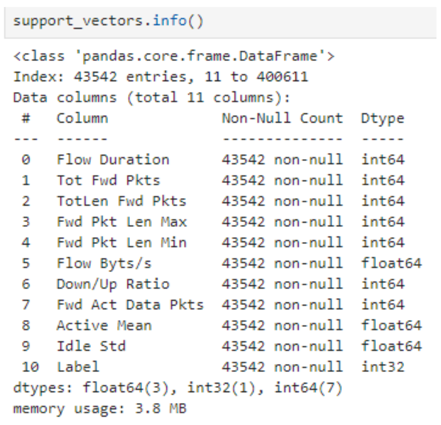


```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.90      0.95     85759
           1       0.91      1.00      0.95     85956

    accuracy                           0.95    171715
   macro avg       0.96      0.95      0.95    171715
weighted avg       0.96      0.95      0.95    171715
```

Fig. 5. Values of estimates of the results of applying the SVM algorithm

After applying the SVM algorithm, we obtained support vectors from the dataset. As a result, we obtained a dataset from the support vectors that contained 10 features and 43542 rows (Fig. 6).



```
support_vectors.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 43542 entries, 11 to 400611
Data columns (total 11 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Flow Duration     43542 non-null  int64
 1   Tot Fwd Pkts      43542 non-null  int64
 2   TotLen Fwd Pkts   43542 non-null  int64
 3   Fwd Pkt Len Max   43542 non-null  int64
 4   Fwd Pkt Len Min   43542 non-null  int64
 5   Flow Byts/s       43542 non-null  float64
 6   Down/Up Ratio     43542 non-null  int64
 7   Fwd Act Data Pkts 43542 non-null  int64
 8   Active Mean       43542 non-null  float64
 9   Idle Std          43542 non-null  float64
 10  Label             43542 non-null  int32
dtypes: float64(3), int32(1), int64(7)
memory usage: 3.8 MB
```

Fig. 6. The list of obtained features of the support vectors and their numer

In the next step of the proposed method, we visualise and examine each feature to determine the distribution of its values relative to the target variable. We define the intervals of the feature values and create the corresponding rules.

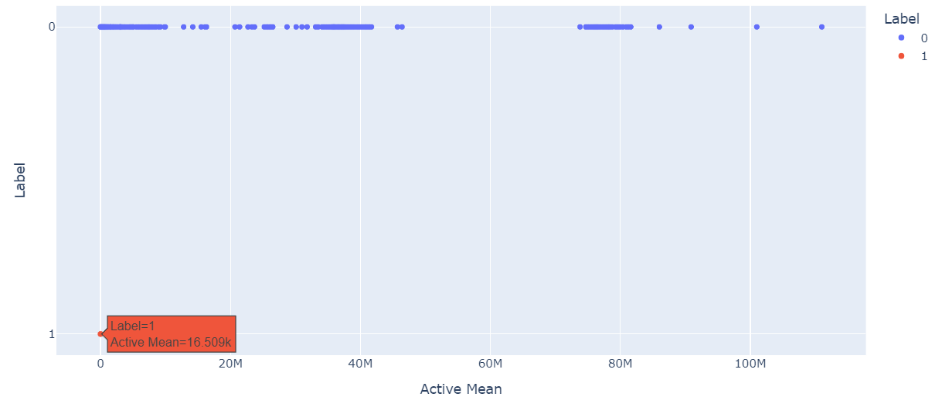
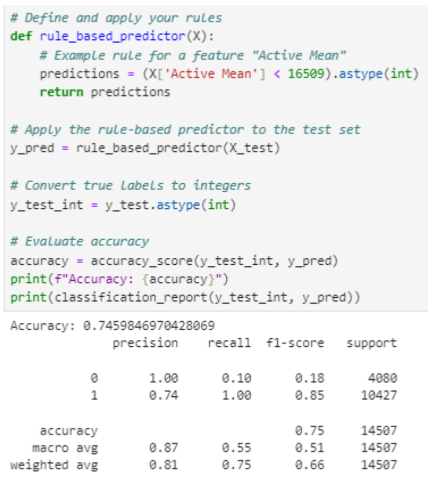


Fig. 7. Distribution of values of Active Mean (Mean time a flow was active before becoming idle) with regard to the target variable

As an example, based on the analysis of the distribution of *Active Mean* values relative to the target variable, we determined the content of the rule and the value of the variable, and obtained estimates of the accuracy of the rule-based classification system (Fig. 8).

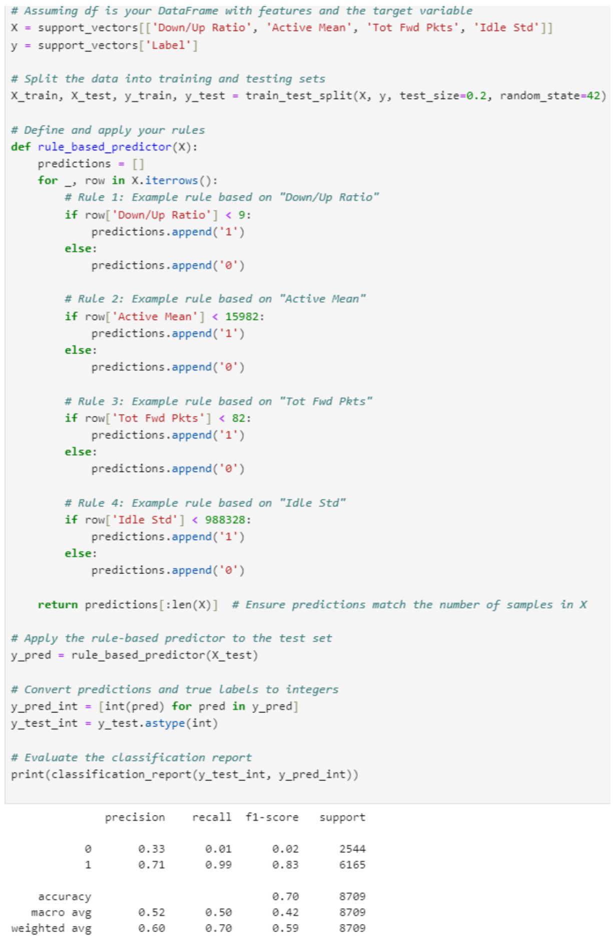


```
# Define and apply your rules
def rule_based_predictor(X):
    # Example rule for a feature "Active Mean"
    predictions = (X['Active Mean'] < 16509).astype(int)
    return predictions

# Apply the rule-based predictor to the test set
y_pred = rule_based_predictor(X_test)

# Convert true labels to integers
y_test_int = y_test.astype(int)

# Evaluate accuracy
accuracy = accuracy_score(y_test_int, y_pred)
print(f"Accuracy: {accuracy}")
print(classification_report(y_test_int, y_pred))
```

```
Accuracy: 0.7459846970428069
              precision    recall  f1-score   support

           0       1.00      0.10      0.18      4080
           1       0.74      1.00      0.85     10427

    accuracy                           0.75     14507
   macro avg       0.87      0.55      0.51     14507
weighted avg       0.81      0.75      0.66     14507
```

Fig. 8. Using the threshold value of the Active Mean feature, taking into account the target variable, to create an appropriate rule and assess the accuracy of detecting malicious processes

If necessary, we combine individual rules into a rule system to identify the malicious process or class of similar processes under consideration. Such an example is shown in Fig. 9.



```
# Assuming df is your DataFrame with features and the target variable
X = support_vectors[['Down/Up Ratio', 'Active Mean', 'Tot Fwd Pkts', 'Idle Std']]
y = support_vectors['Label']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define and apply your rules
def rule_based_predictor(X):
    predictions = []
    for _, row in X.iterrows():
        # Rule 1: Example rule based on "Down/Up Ratio"
        if row['Down/Up Ratio'] < 9:
            predictions.append('1')
        else:
            predictions.append('0')

        # Rule 2: Example rule based on "Active Mean"
        if row['Active Mean'] < 15982:
            predictions.append('1')
        else:
            predictions.append('0')

        # Rule 3: Example rule based on "Tot Fwd Pkts"
        if row['Tot Fwd Pkts'] < 82:
            predictions.append('1')
        else:
            predictions.append('0')

        # Rule 4: Example rule based on "Idle Std"
        if row['Idle Std'] < 988328:
            predictions.append('1')
        else:
            predictions.append('0')

    return predictions[:len(X)]  # Ensure predictions match the number of samples in X

# Apply the rule-based predictor to the test set
y_pred = rule_based_predictor(X_test)

# Convert predictions and true labels to integers
y_pred_int = [int(pred) for pred in y_pred]
y_test_int = y_test.astype(int)

# Evaluate the classification report
print(classification_report(y_test_int, y_pred_int))
```

```
              precision    recall  f1-score   support

           0       0.33      0.01      0.02      2544
           1       0.71      0.99      0.83      6165

    accuracy                           0.70      8709
   macro avg       0.52      0.50      0.42      8709
weighted avg       0.60      0.70      0.59      8709
```

Fig. 9. The example of building a rule-based system and the obtained estimates of the accuracy of detecting malicious processes

4. Discussion

We used support vectors to define the content of the rule system and justify the values of the variables. At the same time, we increase the ability to generalise the classification model by:

- reducing the number of features by applying the Pearson correlation method;
- calculating support vectors using the method of support vectors from the RBF kernel;
- consideration of projections of the distribution of support vector values on the axis of features to determine the content of the rule and the value of the variable;
- determining the content of the rule system to build a rule-based system.

We obtained the corresponding estimates of the accuracy of malware detection by the created rule-based system, but it is necessary to correctly interpret the obtained values of *precision* and *recall* (Fig. 8, Fig. 9).

Accuracy, *precision*, *recall*, and *f1-score* are key metrics used to evaluate the performance of classification models, especially in binary classification tasks (where there are two classes: positive and negative). Here are the main differences between these metrics:

- *accuracy* measures the proportion of correct model answers;
- *precision* measures the proportion of true positive predictions among all positive predictions made by the model;
- *recall* measures the proportion of true positive predictions among all actual positive cases in the dataset.
- *F1-score* shows the harmonic mean of *precision* and *recall*. It provides a single metric that balances *precision* and *recall*.

It should be noted that *precision* focuses on the quality of positive predictions, while *recall* focuses on the coverage of actual positive cases.

Precision is sensitive to false positives, while recall is sensitive to false negatives.

The *F1-score* balances *precision* and *recall* by providing a single metric that takes into account both false positives and false negatives. This is especially useful when classes are unbalanced.

So, *precision* emphasises the correctness of positive predictions, *recall* emphasises the completeness of positive predictions, and *f1-score* provides a balanced measure by taking into account both *precision* and *recall*.

Recall demonstrates the ability of the algorithm to detect this class at all, and *precision* demonstrates the ability to distinguish this class from other classes. The *F1-score* reaches its maximum when *precision* and *recall* are equal to one, and is close to zero if one of the arguments is close to zero.

The purpose of a rule-based system for detecting malicious processes is to detect malicious processes. In our case, *bot* processes. Therefore, our rule-based system detects malicious processes with the values of *precision* = *0.74*, *recall* = *1.00*, *F1-score* = *0.85*, which are quite high.

5. Conclusions

Taking into account the advantages and disadvantages of detection systems based on machine learning methods, in particular, SVM, and rule-based systems for detecting malicious processes in the network traffic of organisations, the paper proposes a methodology for building a hybrid system for detecting such processes.

This methodology has been tested on real data of CSE-CIC-IDS2018 network traffic, which contains characteristics of botnet processes. The basis for justifying the content of the system based on rules and the value of the variables of the rules themselves is an SVM classifier with an RBF kernel.

We have obtained fairly high estimates of the accuracy of detecting malicious processes in network traffic by a hybrid system with high performance. At the same time, we have shown the possibility of building a rule-based system that contains the necessary and sufficient rules to detect certain malicious processes. At the same time, these rules meet the requirements of explainability and interpretability.

We believe that the proposed methodology for building a hybrid system can be used to create automatic systems for detecting malicious processes in the network traffic of organisations.

6. Future work

Ensuring the security of information resources remains a top priority for any organisation around the world. New threats and attack vectors are constantly emerging, making research in this area extremely relevant. Therefore, the creation of effective automatic systems for detecting malicious processes in the network traffic of organisations that meet the requirements of explainability and interpretability is an urgent need of the day. We believe that a promising area for further research is the development of methods for multi-class classification of network traffic for the creation and application of automated intrusion detection systems.

References

[1] A Realistic Cyber Defense Dataset (CSE-CIC-IDS2018). [https://registry.opendata.aws/cse-cic-ids2018] (available: 21.05.2024).

[2] Arrieta A. B. et al.: Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. Information Fusion 58, 2020, 82–115 [https://doi.org/10.1016/j.inffus.2019.12.012].

[3] Barakat N., Bradley A. P.: Rule extraction from support vector machines: A review. Neurocomputing 74(1), 2010, 178–190 [https://doi.org/10.1016/j.neucom.2010.02.016].

[4] Barakat N., Bradley A. P.: Rule Extraction from Support Vector Machines: A Sequential Covering Approach. IEEE Transactions on Knowledge and Data Engineering 19, 2007, 729–741.

[5] Barbado A., Corcho O., Benjamins R.: Rule extraction in unsupervised anomaly detection for model explainability: Application to OneClass SVM. Expert Systems With Applications 189(1), 2022 [https://doi.org/10.1016/j.eswa.2021.116100].

[6] Bologna G, Hayashi Y.: A Rule Extraction Study from SVM on Sentiment Analysis. Big Data and Cognitive Computing 2(1), 2018 [https://doi.org/10.3390/bdcc2010006].

[7] Fung G., Sandilya S., Rao R. B.: Rule extraction from linear support vector machines. Eleventh ACM SIGKDD international conference on Knowledge discovery in data mining (KDD '05). USA, NY, New York, 2005, 32–40 [https://doi.org/10.1145/1081870.1081878].

[8] Hao J., Luo S., Pan L.: Rule extraction from biased random forest and fuzzy support vector machine for early diagnosis of diabetes. Scientific Reports 12(9858), 2022 [https://doi.org/10.1038/s41598-022-14143-8].

[9] Hopgood A. A.: Intelligent Systems for Engineers and Scientists: A Practical Guide to Artificial Intelligence (4th ed.). CRC Press 2022 [https://doi.org/10.1201/9781003226277].

[10] Jiawei Z., Hongyang J., Ning Z.: Alternate Support Vector Machine Decision Trees for Power Systems Rule Extractions. TechRxiv. 11, 2022 [https://doi.org/10.36227/techrxiv.20445150.v1].

[11] Kambourakis G. et al.: Botnets: Architectures, Countermeasures, and Challenges (1st ed.). CRC Press, 2019 [https://doi.org/10.1201/9780429329913].

[12] Kašćelan L., Kašćelan V. Jovanović M.: Hybrid support vector machine rule extraction method for discovering the preferences of stock market investors: Evidence from Montenegro. Intelligent Automation & Soft Computing 21(4), 2014, 503–522 [https://doi.org/10.1080/10798587.2014.971500].

[13] Martens D., Baesens B. B., Van Gestel T.: Decompositional Rule Extraction from Support Vector Machines by Active Learning. IEEE Transactions on Knowledge and Data Engineering 21(2), 2009, 178–191 [https://doi.org/10.1109/TKDE.2008.131].

[14] Newman J.: A Taxonomy of Trustworthiness for Artificial Intelligence. CLTC. White Paper. January 2023. [https://cltc.berkeley.edu/publication/a-taxonomy-of-trustworthiness-for-artificial-intelligence/] (available: 21.05.2024).

[15] Núñez H., Angulo C., Català A.: Rule extraction from support vector machines. European Symposium on Artificial Neural Networks (ESANN'2002). Belgium, Bruges, 2002, 107–112.

[16] Núñez H., Angulo C., Català A.: Rule-Based Learning Systems for Support Vector Machines. Neural Process Lett 24, 2006, 1–18 [https://doi.org/10.1007/s11063-006-9007-8].

[17] Shigeo Abe: Support Vector Machines for Pattern Classification. Second Edition. Springer-Verlag London Limited 2005, 2010. [https://doi.org/10.1007/978-1-84996-098-4].

[18] Tian Y., Shi Y., Liu X.: Recent Advances on Support Vector Machines Research. Technological and Economic Development of Economy 18(1), 2012, 5–33 [https://doi.org/10.3846/20294913.2012.661205].

[19] Yang S. X., Tian Y. J., Zhang C. H.: Rule Extraction from Support Vector Machines and Its Applications. IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology. France, Lyon, 2011, 221–224 [https://doi.org/10.1109/WI-IAT.2011.132].

[20] Zhu P., Hu Q.: Rule extraction from support vector machines based on consistent region covering reduction. Knowledge-Based Systems 42, 2013, 1–8 [https://doi.org/10.1016/j.knosys.2012.12.003].

Prof. Halyna Haidur
e-mail: gaydurg@gmail.com

Doctor of Technical Sciences, professor, Head of Department of Information and Cyber Security, State University of Information and Communication Technologies, Kyiv, Ukraine.
Interested in: cybersecurity technologies, application of machine learning in cybersecurity, methods and tools for detecting anomalies in network traffic.

https://orcid.org/0000-0003-0591-3290

Ph.D. Sergii Gakhov
e-mail: gakhovsa@gmail.com

Candidate of Military Sciences, associate professor of Department of Information and Cyber Security, State University of Information and Communication Technologies, Kyiv, Ukraine.
Interested in: network security, anomaly detection, machine learning in cybersecurity.

https://orcid.org/0000-0001-9011-8210

Dmytro Hamza
e-mail: supprius@gmail.com

Postgraduate student of Department of Information and Cyber Security, State University of Information and Communication Technologies, Kyiv, Ukraine.
Interested in: machine learning in cybersecurity.

https://orcid.org/0009-0005-0947-2420