Analysis of ML methods and algorithms for the detection of advanced attacks (APT)

INNOvative SOC/INNOSOC project

Contents

Introduction 3
Experiments and results 6
Datasets 6
DAPT2020 6
Algorithms and methods used 8
Random forest 8
SVM 12
Logistic regression 13
Settings and parameters of the selected methods 15
Logistic regression 15
Random forest 15
SVM 16
Evaluation of experiments 16
Confusion matrix 17
Accuracy 18
Precision 18
Recall 18
F1-Score 19
Results 19
Summary 23

Introduction

An advanced persistent threat (APT) is a form of cyber threat posed by well-funded organizations, often with the aim of obtaining key information from the target organization. An APT threat differs from conventional and multi-stage attacks and is defined by a combination of three words:

- **Advanced** – Advanced – An advanced threat actor is technically proficient, experienced and capable of discovering or introducing new vulnerabilities into a system and developing new ways to exploit these vulnerabilities. Attackers even resort to various attacks, such as a zero-day exploit, planted insiders or supply chain exploitation, when the target's defences require it.
- **Persistent** – Persistent – A persistent threat is anyone who demonstrates the determination to attack until the objective is achieved, is skilled and experienced in adapting to hostile territory in order to remain undetected, and has the resources to evolve their TTPs to defeat defensive strategic or tactical countermeasures. These resources include various assets and are not specifically limited to financial means. Human capital, for example, is a highly relevant resource. A persistent threat will have access to personnel capacities for every operation it initiates. They work predominantly in teams, as the scope and range of their activities are sufficiently extensive. Other assets are infrastructure, time and financial resources.
- **Threat** – Threat – A threat refers to individuals (e.g. persons, groups and organizations) who have the motivation, capability and opportunity to engage in malicious acts. The key defining feature of a threat is its malicious intent, which is the driving force of its motivation.

When considering APTs, it is important to realize that they are people, as this term was specifically designed to distinguish them from simple malware. As Richard Bejtlich states: "Threat means that the adversary is not a piece of mindless code." Technically, there is no way to capture human creativity or imagination. Therefore, we cannot assume that we are able to predict what attack an APT will carry out. Research often regards APT attacks as targeted multi-stage attacks. However, the constraint that a multi-stage attack must consist of at least two atomic or minor events is not mandatory for an APT attack. If an attacker needs only one action to achieve the objective, they will use it. The path of least resistance, as part of the term Advanced in the definition, describes their ability to exploit the entire spectrum of intrusions. For example, if there is a known vulnerability with the corresponding exploit material (*exploit material*), an APT *can* and *will* exploit it to achieve its objectives.

A multi-stage attack may also hit one specific target. Detection systems therefore have difficulty distinguishing the threats behind an attack.

APTs seek to remain undetected and therefore use concealment techniques (*concealment*), one of the most frequently used being the *living-off-the-land* technique. These techniques involve disguising malicious activity as seemingly harmless programs that use tools already present on users' computers. These tools are often part of the operating system or other software installed by the user. This type of attack is particularly difficult to detect, because attackers usually do not create new malicious files on the target computers and regular antivirus scans may not identify them.

These techniques are not taken into account in conventional cyber threat intelligence (CTI), because they require a higher level of sophistication from the attacker and cannot be compared with known procedures. This fact highlights the importance of taking these advanced tactics into account in security measures and of continuously updating CTI, so that we stay one step ahead of the changing threat landscape.

Nevertheless, an APT can and will use known attack patterns, such as replay attacks (*replay attacks*) or code injection (*injection*), via known vulnerabilities. Established rule-based systems and data-driven approaches for specific TTPs are a partial solution.

In its worst manifestation, an APT can be described as a complex adversary with a broad spectrum of attacks, characterized by the DoD (Department of Defense) and sometimes called integrated threats. This characteristic means that they do not look for opportunities to strike – they use political, military, intelligence or economic resources to enable their operations.

Historical examples of APT attacks that are particularly interesting are the Stuxnet attack, which was used to destroy uranium enrichment centrifuges in Natanz in Iran, Industroyer, which caused a power outage in Ukraine in 2016, and TRITON, which attacked safety instrumented systems in the oil industry. Among other APT attacks, a significant pillar in the evolution of APTs is the shift from predominantly espionage to destructive attacks.

In addition to stealing information and manipulating or even destroying targets, an APT may also seek to install *backdoor* software in order to gain easy access at strategically advantageous times. In addition to the use of various tactics, techniques and procedures (TTPs) (*Tactics, Techniques, and Procedures*), the types of attacks also differ in terms of attacker, target, purpose and approach. The table below shows the distinction between the types of attacks.

	Conventional attack	Multi-stage attack	APT attack
Attacker	predominantly an individual	criminally motivated groups	Highly organized, sophisticated, determined and well-funded group
Target of the attack	Unspecified, mostly individual systems	Domain-specific organizations	Specific targets of strategic value
Purpose of the attack	Financial gain, demonstration of capabilities	Financial gain, demonstration of capabilities	Competitive advantages, strategic advantages
Attack approach	One-off, short-lived *smash-and-grab* operation (smash and grab)	One-off, conspicuous and overt operation lasting from several hours to weeks	Repeated attempts, a low-and-slow (*low-and-slow*) approach, adaptation to long-term defensive resistance and frequent concealment until the backdoor (*backdoor*) has strategic use

Tab. 1: Comparison of conventional, multi-stage and APT attacks

Various methodologies can be used in the area of detection, the most common approach being to differentiate tools according to their detection objectives. In principle, the following methods are distinguished:

- **Rule**-based methods (*rule-based*): This is an attack detection method that uses predefined rules and patterns to identify suspicious activity. These rules and patterns are usually based on knowledge of known attacks or of behaviour patterns that are considered normal in the system. This category is therefore also referred to as signature-based detection or knowledge-based detection, which can be further divided into signature matching, state transition analysis and expert systems.
- **Anomaly**-based methods (*anomaly-based*): Anomaly detection identifies unusual patterns or instances in data that do not correspond to the expected behaviour. The key difference is that anomaly detection does not distinguish between known and unknown attacks. Since anomaly detection usually relies on system behaviour to detect deviations, these systems are less widespread than rule-based systems, especially considering that the learning process must precede the machine learning variants.
- **Statistical** methods: This group includes all methods based on statistical analysis for the purposes of prediction and detection. Most statistical attack detection methods use probabilistic models and training data to monitor specific behaviour.
- **Classification** methods: The classification group includes methods that also use machine learning techniques, but with supervised learning, in which they learn

benign and malicious behaviour and then label new activities according to the learned classes. In addition, multi-class labelling (*labelling*) can achieve more accurate detections if the training dataset is appropriately labelled and contains a representative sample of attacks from each class. New behaviour that resembles neither previous benign nor previous malicious patterns cannot be adequately handled.

- **Human knowledge**: Detection systems are placed in the human knowledge category when they make use of strong domain expertise or rely on CTI.

Experiments and results

Datasets

DAPT2020

The DAPT2020 dataset (Dataset for Advanced Persistent Threats (APT) 2020) is a cybersecurity dataset specifically designed for research in the area of detection and analysis of advanced persistent threat (APT) attacks. It was created and published in 2020 by the Cyber Threat Intelligence (CTI) laboratory at the University of New South Wales (UNSW) in Canberra, Australia.

The primary objective was to address the lack of realistic, up-to-date and high-quality datasets that would capture the multi-stage, stealthy and long-term nature of APT attacks. Unlike datasets containing simple exploits or short-term attacks, DAPT2020 focuses on simulating the entire cyber attack chain (*cyber kill chain*), which represents sophisticated APT campaigns, and provides data for developing detection systems for lateral movement (*lateral movement*), privilege escalation (*privilege escalation*), data exfiltration (*data exfiltration*) and command and control (C2) activities (*command & control*).

The dataset was generated in a controlled sandbox-type test environment designed to emulate a real enterprise network, not from a live production system.

- **Network topology**: A small enterprise network was simulated with multiple operating systems (Windows 10, Windows Server, Ubuntu) and standard services (Active Directory, web servers, file shares).
- **Attack simulation**: The researchers executed real malware and APT attack tools according to the documented tactics, techniques and procedures (TTPs) (Tactics, Techniques, and Procedures) of known APT groups.
- **Benign background traffic**: To create a realistic mix, regular user activities (web browsing, e-mail, file transfer, etc.) were also generated over several days using automated scripts and volunteers.

The dataset was created by simulating the behaviour of APT attacks in a cloud network over five days:

- **Day 1**: Normal traffic - ping, web browsing, file operations, ...
- **Day 2**: Reconnaissance (*reconnaissance*) - nmap, sqlmap, vulnerability scanning (*vulnerability scanning*), ...
- **Day 3**: Establishing a foothold (*foothold*) - reverse shells (*reverse shells*), SQL exploitation (*SQL exploitation*), authentication bypass (*authentication bypass*)
- **Day 4**: Lateral movement (*lateral movement*) - internal network scanning, exploiting vulnerabilities (*exploiting vulnerabilities*)
- **Day 5**: Data exfiltration (*data exfiltration*) - FTP transfers, uploads to Google Drive, C&C communication

The dataset contains 79 network flow parameters/features (features), including flow duration, packet sizes, inter-arrival times, TCP flags, header lengths, byte/packet transfer rates and various statistical indicators.

The dataset is obtained from multiple sources, which provides a holistic view that is crucial for APT detection. The DAPT2020 dataset includes:

- **Full packet capture (PCAP)**: The complete network traffic from the test environment, which enables in-depth protocol inspection.
- **Network flow data (NetFlow)**: Summarized connection logs, which are less detailed but efficient for analysis.
- **System logs**:
    - Windows event logs: Security, system and application logs, which are crucial for detecting lateral movement (*lateral movement*), privilege escalation (*privilege escalation*) and persistence (*persistence*).
    - Sysmon logs: Detailed process creation, network connection and file modification events (invaluable for forensic tracking).
- **Host artefacts** (*host-based artifacts*):
    - Memory dumps: For malware analysis.
    - Disk images: For forensic investigation.
    - Process listings and registry snapshots.
- **Alert and detection logs**: Outputs from security tools such as Suricata (IDS) and Elastic Security (SIEM) deployed during the experiment.

The dataset contains two distinct, interlinked APT campaigns inspired by real-world groups (e.g. similar to APT29/Cozy Bear or APT32/OceanLotus):

1. **Campaign 1**: Focuses on data theft and leakage. It includes an initial compromise attack, lateral movement to a database server, data staging and leakage via covert channels.

2. **Campaign 2**: Focuses on ransomware deployment. It includes establishing a foothold (foothold), escalating privileges (*escalating privileges*), disabling defences (*disabling defenses*) and deploying ransomware in the network.

These campaigns include phases such as reconnaissance (*reconnaissance*), initial access (e.g. *spear-phishing*), execution (*execution*), persistence (*persistence*), lateral movement (*lateral movement*) and exfiltration (*exfiltration*).

The dataset has the following key properties:

- It uses modern TTPs and malware samples from the 2019 – 2020 time frame.
- It captures attacks that unfold over the course of days, not seconds.
- The attacks are carefully labelled with precise timestamps and mapped to the MITRE ATT&CK framework, which makes them excellent for supervised machine learning and TTP-based research.
- The synchronous collection of network (network) and host (host) records enables correlation across sources, which is a necessity for APT hunting (APT hunting).

The dataset is publicly available for research purposes.

https://www.kaggle.com/datasets/sowmyamyneni/dapt2020

More detailed information about the dataset can be found in the supporting document DAPT_at_MLHat2020.pdf.

Algorithms and methods used

Random forest

Decision trees

A decision tree is a non-parametric algorithm that is used for both classification and regression tasks. It has a hierarchical tree structure consisting of a root node, branches,

internal nodes and leaf nodes. As can be seen from the diagram below, a decision tree starts with a root node, which has no incoming branches.

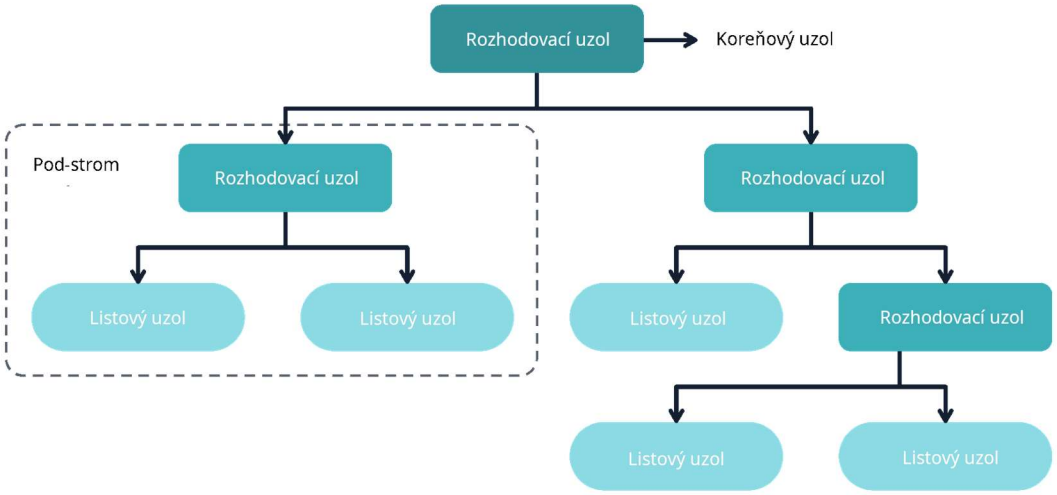


Fig. 1: Decision tree

The outgoing branches from the root node then feed into the internal nodes, also known as decision nodes. Based on the available features, both types of nodes perform evaluations in order to form homogeneous subsets, which are denoted by leaf nodes or terminal nodes. The leaf nodes represent all possible outcomes within the dataset. As an example, imagine that we are trying to assess whether or not we should go and play golf; we can use the following decision rules to make the choice:

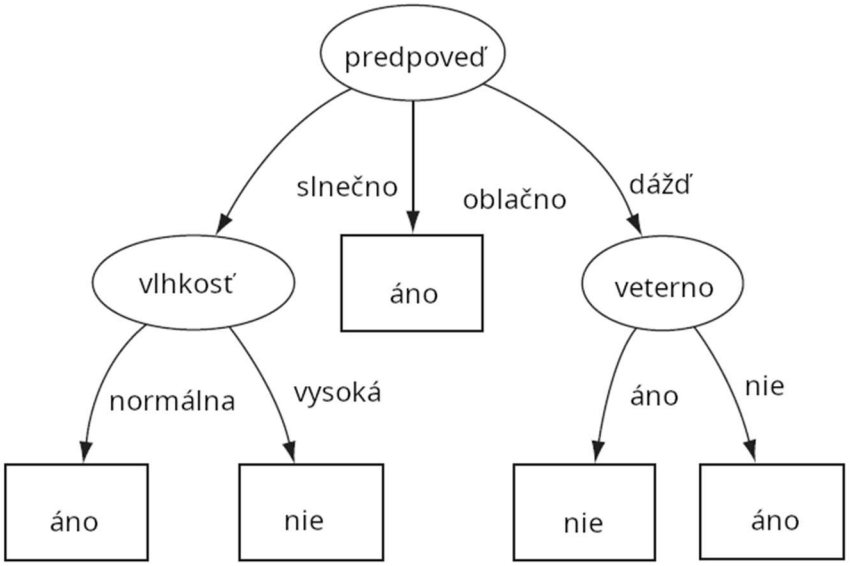


Fig. 2: Decision tree for the Golf dataset

This type of flowchart structure also creates an easily digestible representation of decision-making, which allows different groups across the entire organization to better

understand why a decision was made. Decision tree learning uses a divide-and-conquer strategy by performing a greedy search to identify the optimal split points in the tree. This splitting process is then repeated in a top-down, recursive manner until all or most of the records have been classified under specific class labels. Whether all data points are classified as homogeneous sets depends to a large extent on the complexity of the decision tree. Smaller trees are more easily able to achieve pure leaf nodes – i.e. data points in a single class. However, as the tree grows, it becomes increasingly difficult to maintain this purity, and it usually results in too little data falling within a given subtree. When this happens, the phenomenon is known as data fragmentation, and it can often lead to overfitting. As a result, decision trees favour small trees, which is in line with the principle of parsimony according to "Occam's razor", meaning that "entities should not be multiplied unless necessary". In other words, decision trees should increase their complexity only when necessary, because the simplest explanation is often the best. To reduce complexity and prevent overfitting, pruning is usually used; This is a process that removes branches that split on features of low importance. The fit of the model can then be evaluated through a cross-validation process.

Random Forest

Random forests (*random forest*) are ensemble learning methods (*ensamble learning*) that are used for either classification or regression tasks. Random forests consist of multiple decision trees that are combined into a unanimous decision or classification (figure below).

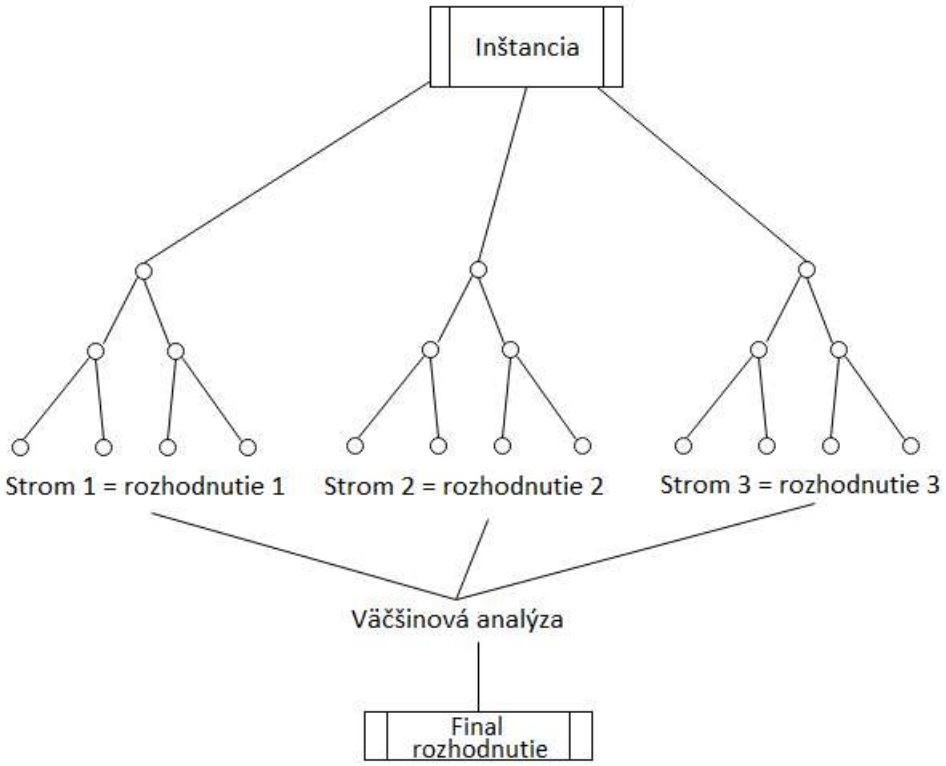


Fig. 3: Random forest – operating principle

Random forests are better than ordinary decision trees because they do not cause overfitting of the data and subsequent overtraining of the model. This is achieved through the use of methods that employ random subspaces (for example *bagging*) to reduce the correlation between individual trees. The elementary algorithms/trees are trained on different subsets of the feature space, which are also randomly selected. An ensemble of decision tree models using the random subspace method can be built using the following algorithm.

For N training objects and a number of features D , we proceed as follows:

1. The parameter L is chosen as the number of individual trees in the ensemble (*ensemble*).
2. For each individual l-th tree, $d_l < D$ is selected as the number of features for l. Typically, only one value is used for all trees.
3. For each tree, a training subset X_n is created using the bootstrap method.

The decision trees are created from the X_n samples as follows:

1. Random features d_l are selected from the source; the optimal split of the training set then restricts its search to them.
2. According to the given criterion, the best attribute is selected and a split in the tree is made according to it.
3. The tree is grown until no more than n_{min} objects remain in each leaf, until a certain tree height is reached, or until the training set is exhausted.

In order to apply the model ensemble to a new data object, it is necessary to combine the results of the individual models by majority voting or by combining *a posteriori* probabilities. An example of the final classifier is as follows:

$$a(x) = \frac{1}{L}\sum_{i=1}^{L} l_i(x)$$

The basic parameters of the *random forest* method include:

- number of trees: The more trees, the better the quality of the model, but the training time and the workload of the algorithm also increase proportionally. Often, as the number of trees increases, the quality on the training set rises (it may even reach 100% accuracy), but the quality on the test set is asymptotic (so the minimum required number of trees can be estimated).
- number of features for selecting the tree split: As the number of features increases, the time needed to build the random forest also increases and the trees are more uniform than before. In classification problems, the number of attributes is often chosen as the value $\sqrt{D}$ and for regression problems as the value $D/3$.
- maximum tree depth: The smaller the depth, the faster the algorithm is built and will operate. As the depth increases, the quality during training increases

dramatically. The quality may also increase on the test set. It is recommended to use the maximum depth (except in cases where there are too many training objects and we obtain very deep trees whose construction takes considerable time). When shallow trees are used, changing the parameters associated with limiting the number of objects in a leaf and at splitting does not lead to a significant effect (the leaves are already large). The use of shallow trees is recommended for tasks with a large number of noisy (*noisy*) or outlying objects (*outliers*), which represent random or irrelevant data.

- impurity criterion (*impurity criterion*): This criterion is used to select the feature (decision rule) for branching. For regression problems, the mean squared error MSE (*Mean Squared Error*) or the mean absolute error MAE (Mean Absolute Error) is usually used. For classification problems, it is the Gini criterion, entropy or classification error. The balance and depth of the trees may vary depending on the specific impurity function chosen.

SVM

The support vector machine is a classification and regression method proposed in the 1990s by V. Vapnik, which essentially determines a boundary curve separating 2 different classes. The main idea is to find a hyperplane that would optimally divide the input data in the feature space. For a given set of training data belonging to 2 classes (x_1, y_1), …,(x_l, y_l), where $x_i \in R_n$ and $y_i \in \{-1, 1\}$, we therefore try to find a hyperplane $wx + b = 0$ such that we can separate the data. Of all the hyperplanes determined by w and b, there is only one that maximizes the distance between the nearest points of these classes and the hyperplane - the optimal hyperplane, and the following holds:

$$d(w,b) = \min_{\{x_i | y_i = 1\}} \frac{wx_i + b}{|w|} - \max_{\{x_i | y_i = -1\}} \frac{wx_i + b}{|w|}$$

and the corresponding minimum and maximum values are -1 and +1, respectively

$$d(w,b) = \left| \frac{1}{\mathbf{w}} - \frac{-1}{\mathbf{w}} \right| = \frac{2}{|\mathbf{w}|} \ .$$

By minimizing the expression $|w|^2 / 2$ subject to the condition $y_i (wx_i + b) \geq 1$, we obtain the maximum distance between the 2 classes for the given pair of hyperplanes. The solution of the SVM optimization problem is given by the saddle point (*saddle point*) of the Lagrangian function:

$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i \left[y_i \left(w x_i + b \right) - 1 \right] ,$$

where α_i is the Lagrange multiplier. While $L(w, b, \alpha)$ decreases with w and b, the derivatives of $L(w, b, \alpha)$ are equal to zero for all $\alpha_i \geq 0$. The solution is given by

$$\overline{w} = \sum_{i=1}^{l} \overline{\alpha}_i y_i x_i \qquad \overline{b} = -\frac{1}{2} \overline{w} \left[x_r + x_s \right] ,$$

where x_r and x_s are support vectors belonging to the class $y_r = 1$ and $y_s = -1$, respectively. Data in the original feature space are not separable by any linear hyperplane, but they may be linearly separable in a non-linearly mapped feature space, which differs from the original one and usually has a higher dimension. This non-linear mapping $K(.,.)$ is performed using kernel functions (*kernel functions*):

$$f(x) = \operatorname{sgn}\left(\sum_{i=1}^{l} \overline{\alpha}_i y_i K(x_i, x) + b \right) \sum_{i=1}^{N} \alpha_i t_i K(x, x_i) + b ,$$

where yi are the target values, $\sum_{i=1}^{l} \overline{\alpha}_i y_i = 0$ and $\overline{\alpha}_i > 0$. The kernel $K(.,.)$ is constructed so as to satisfy specific properties depending on the given problem, and it can be written:

$$K(x, y) = \Phi(x)\Phi(y) ,$$

where $\Phi(x)$ is the mapping of the input feature space into a new space with a potentially infinite dimension. There are 3 kernel functions providing non-linear mapping:

- polynomial $K(x, y) = (xy + 1)^z$ – the parameter z represents the degree of the polynomial
- Gaussian radial basis functions (Radial Basis Functions RBF) $K(x, y) = exp\{-[(x-y)^2 / 2\sigma^2]\}$ – where the parameter σ is the standard deviation of the Gaussian function
- MLP function $K(x, y) = tanh(scale(xy) - offset)$ – *scale* and *offset* are given parameters
- Sigmoid $K(x, y) = tanh(\beta xy + \theta) - \beta, \theta$ are given parameters.

Logistic regression

Logistic regression, also known as logistic regression analysis, is a statistical analysis method that predicts the corresponding binary outcomes based on the relationship between an independent variable and a dependent variable, and it can therefore be used for two-class classification. Classification is essentially a regression problem with a discrete output - the logistic regression algorithm reflects this nature.

Let X denote the input space of the classification and $C = \{0, 1\}$ denote the output space of the two-class classification. For input data $\boldsymbol{x} \in X$, the logistic regression $P(c|\boldsymbol{x})$ for predicting their category as $c \in C$ is given as follows:

$$P(c|x) = \sigma(z) = \frac{1}{1+e^{-z}} = \frac{1}{1+\exp(-z)},$$

where $\sigma(\cdot)$ is referred to as the logistic function, also known as the sigmoid or sigmoidal function, which is also used as one of the activation functions in artificial neural networks.

Logistic regression has the following property, i.e. $P(c = 0|\boldsymbol{x}) = 1 - P(c = 1|\boldsymbol{x})$. This is because the probability distribution of a two-class classification should satisfy the condition $P(c = 0|\boldsymbol{x}) + P(c = 1|\boldsymbol{x}) = 1$. Also, let $z = \boldsymbol{w}^{\mathsf{T}}\boldsymbol{x} - b$, from which the following is obtained:

$$P(c=1|x) = \sigma\left(w^{\mathsf{T}}x - b\right) = \frac{1}{1+\exp\left(-\left(w^{\mathsf{T}}x - b\right)\right)},$$

$$P(c=0|x) = 1 - \sigma\left(w^{\mathsf{T}}x - b\right) = 1 - \frac{1}{1+\exp\left(-\left(w^{\mathsf{T}}x - b\right)\right)}$$

$$= \frac{\exp\left(-\left(w^{\mathsf{T}}x - b\right)\right)}{1+\exp\left(-\left(w^{\mathsf{T}}x - b\right)\right)}.$$

A schematic diagram of the logistic regression classification algorithm is shown in the following figure.

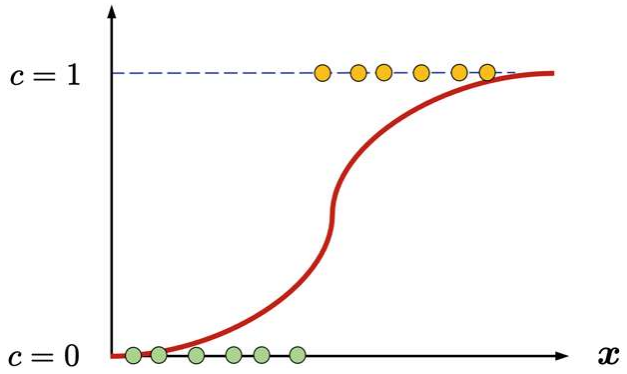


Fig. 4: Principle of the logistic regression classification algorithm

The logistic regression method described above is applicable only to two-class classification. Later, some extended versions of logistic regression appeared for multi-class classification problems, including multinomial logistic regression, ordered logistic regression and conditional logistic regression.

Settings and parameters of the selected methods

Logistic regression

- C:1.0
- class_weight:None
- dual:False
- fit_intercept:True
- intercept_scaling:1
- l1_ratio:None
- max_iter:1000
- multi_class:deprecated
- n_jobs:None
- penalty:l2
- random_state:42
- solver:lbfgs
- tol:0.0001
- verbose:0
- warm_start:False

Random forest

- bootstrap:True
- ccp_alpha:0.0
- class_weight:None
- criterion:gini
- max_depth:None
- max_features:sqrt
- max_leaf_nodes:None
- max_samples:None
- min_impurity_decrease:0.0
- min_samples_leaf:1
- min_samples_split:2
- min_weight_fraction_leaf:0.0
- monotonic_cst:None
- n_estimators:100
- n_jobs:None
- oob_score:False
- random_state:42
- verbose:0
- warm_start:False

SVM

- C:1.0
- break_ties:False
- cache_size:200
- class_weight:None
- coef0:0.0
- decision_function_shape:ovr
- degree:3
- gamma:scale
- kernel:rbf
- max_iter:-1
- probability:False
- random_state:42
- shrinking:True
- tol:0.001
- verbose:False

Evaluation of experiments

Several techniques have been proposed over time as quantitative measures for assessment or evaluation:

- Coverage (*Coverage*) may refer to vulnerability coverage or attack coverage, which uses existing lists of possible attacks or vulnerabilities to determine which are detected and which are not.
- Probability of detection and false alarms (*Probability of detection and false alarms*) refer to the false and true alarms generated within a given time frame. True alarms refer to alarms that are attributable to an attack, whereas false alarms are not caused by an attack. This can be represented as a receiver operating curve ROC (*Receiver Operating Curve*) as an aggregation. The percentages of true and false alarms are thus contrasted and represent the detection system in different operating modes.
- Resistance (*Resistance*) to attacks directed at intrusion detection systems IDS (I*ntrusion Detection Systems*): According to this metric, various attacks on the detection system are measured and it is checked whether the detection system continues to function correctly. These attacks may be, for example, overtly deliberate intrusion attacks. Bandwidth also belongs to this category, but it indicates what amount of load is generally acceptable.
- The ability to correlate events describes the ability of the detection system to correlate attacks from different sources. Conceivable categories are source data correlation, alarm correlation and any correlation.

- The ability to identify an attack means how well the detection system assigns a label to an alarm.
- The ability to determine the success of an attack: Another difference is how well the detection system can distinguish between successful and unsuccessful attacks.
- Other metrics (unrelated to detection performance) are ease of use, maintenance, configuration issues, resource requirements, availability and quality of support.

The following metrics were used to evaluate the experiments.

Confusion matrix

The confusion matrix is one of the most popular ways of evaluating any classification model. Although the matrix itself is not a metric, the matrix representation can be used to define various metrics, all of which become important in some specific case or scenario. A confusion matrix can be created for binary classification as well as for a classification model with multiple classes.

A confusion matrix is created by comparing the predicted class label of a data point with its actual class label. This comparison is repeated for the entire dataset (dataset), and the results of this comparison are compiled in a matrix or tabular format. This resulting matrix represents the confusion matrix (fig. below).

		Skutočná trieda	
		Kladná	Záporná
Predikovaná trieda	Kladná	Počet správne kladných (TP)	Počet nesprávne kladných (FP)
	Záporná	Počet nesprávne záporných (FN)	Počet správne záporných (TN)

Fig. 5: Confusion matrix

In general, we usually have 2 classes – the positive class and the negative class. Based on this structure, we can clearly see four terms of importance.

• True Positive (TP) – Number of true positives: This is the sum of the total number of examples (data points from the dataset) from the positive class where the actual class label was equal to the predicted class label, i.e. the total number of cases in which we correctly predicted the positive class label using our model.

• False positive (FP) – Number of false positives: This is the number of total cases from the negative class that the model incorrectly classified by predicting them as positive. Hence the name, false positive.

• True Negative (FN) – Number of false negatives: This is the number of the total number of examples from the negative class where the actual class label was equal to the predicted class label, i.e. the total number of cases in which we correctly predicted the negative class label using our model.

• False negative (FN) – Number of false negatives: This is the number of the total number of cases from the positive class that our model incorrectly classified by predicting them as negative. Hence the name, false negative.

Accuracy

This metric is one of the most popular ways of measuring or evaluating the effectiveness of any classifier. It is defined as the overall accuracy or the proportion of correct predictions of the model. The formula for calculating it from the confusion matrix is as follows:

$$A = \frac{TP + TN}{TP + FP + TN + FN}$$

Precision

Precision (precision), also known as positive predictive value, is another metric that can be derived from the confusion matrix. This metric is defined as the number of predictions that are actually correct or relevant out of all predictions based on the positive class. The formula for this metric is as follows:

$$P = \frac{TP}{TP + FP}$$

A model with a high precision value identifies a higher proportion of the positive class compared with a model with lower precision. This metric becomes important in cases where we are more interested in finding the maximum number of positive classes, even if the overall accuracy decreases.

Recall

Recall (recall) is a model metric for identifying the percentage of relevant data points. It is defined as the number of instances of the positive class that were correctly predicted. It is also referred to as the coverage rate or sensitivity. The formula for calculating this metric is as follows:

$$R = \frac{TP}{TP + FN}$$

This metric becomes an important indicator of classifier effectiveness in scenarios in which we want to capture the largest possible number of cases of a particular class, even if this causes an increase in false detections/predictions. As an example, in the case of bank fraud, a model with a high recall value will provide us with a higher number of potential fraud cases. However, it will equally increase the triggering of alarms for most suspicious cases.

F1-Score

There are cases in which we want to achieve a balanced optimization of precision and recall. The F1-Score is a metric that is the harmonic mean of the precision and recall metrics and helps us to optimize the classifier for balanced precision and recall of prediction. The formula for calculating the F1-Score is as follows:

$$F = \frac{2.P.R}{P+R}$$

Results

Table 3 gives a clear overview of the test results for the selected classification methods.

Method	Accuracy	Precision	Recall	F1-score	Processing time (sec.)
Log.Regression	0.9018	0.9005	0.9018	0.8970	0.17
Random Forest	**0.9842**	**0.9842**	**0.9842**	**0.9842**	0.19
SVM	0.9269	0.9312	0.9269	0.9212	48.91

Tab. 3: Comparison of the individual classification methods

The following figures show the confusion matrix for each of the tested methods.

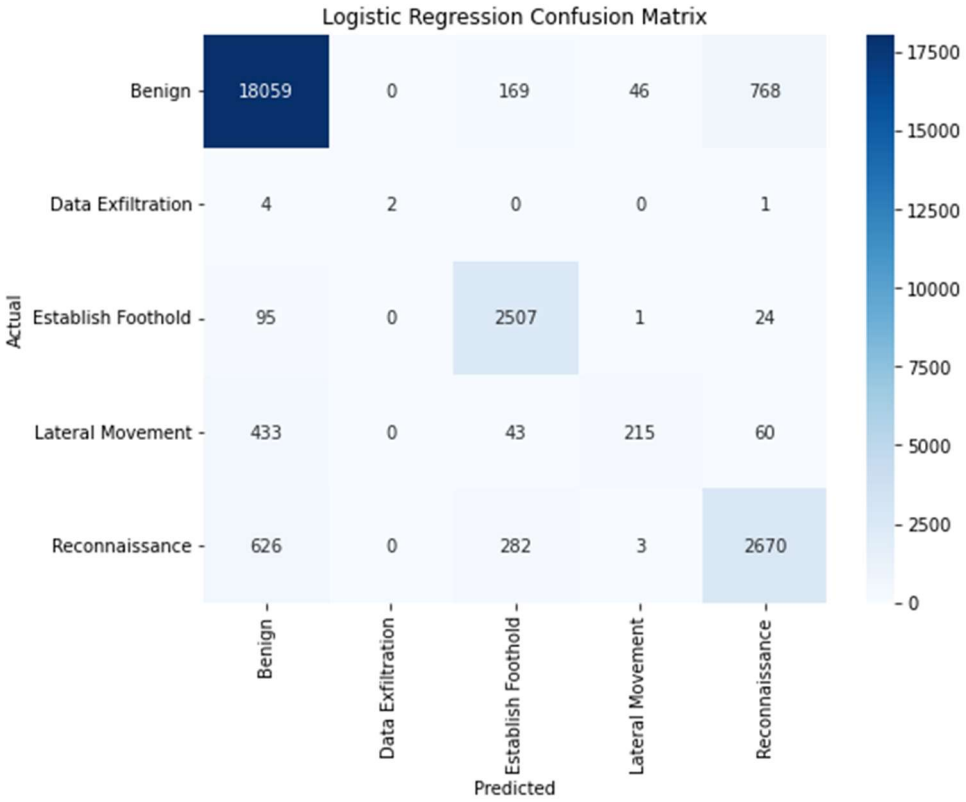


Fig. 6: Confusion matrix for logistic regression

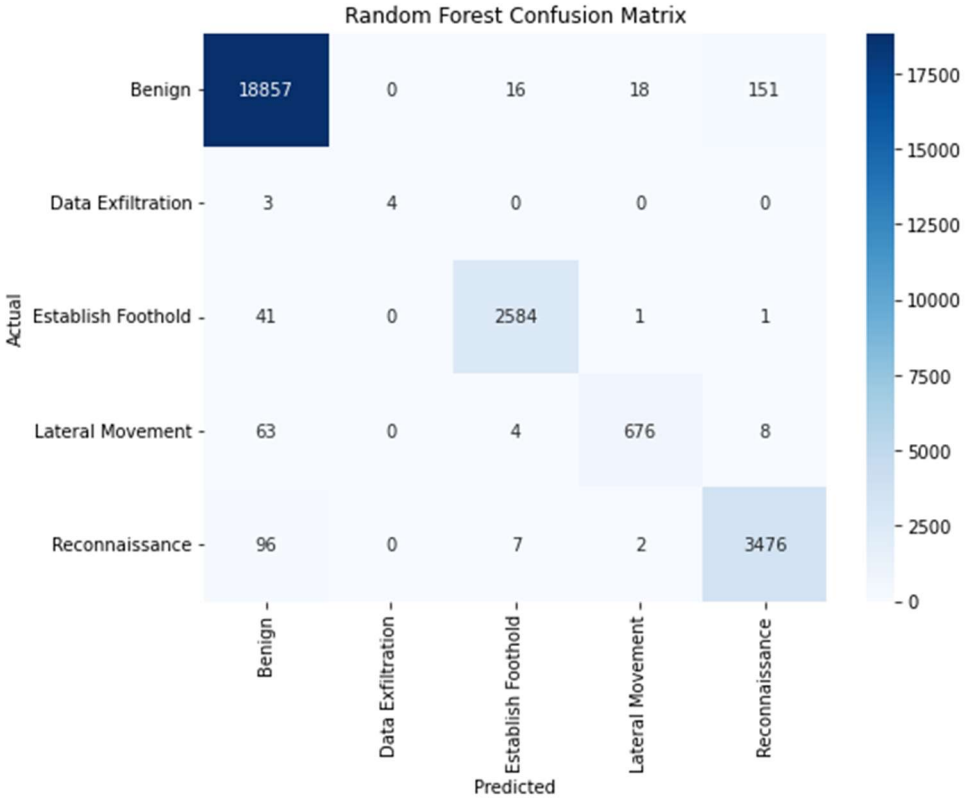


Fig. 7: Confusion matrix for random forest

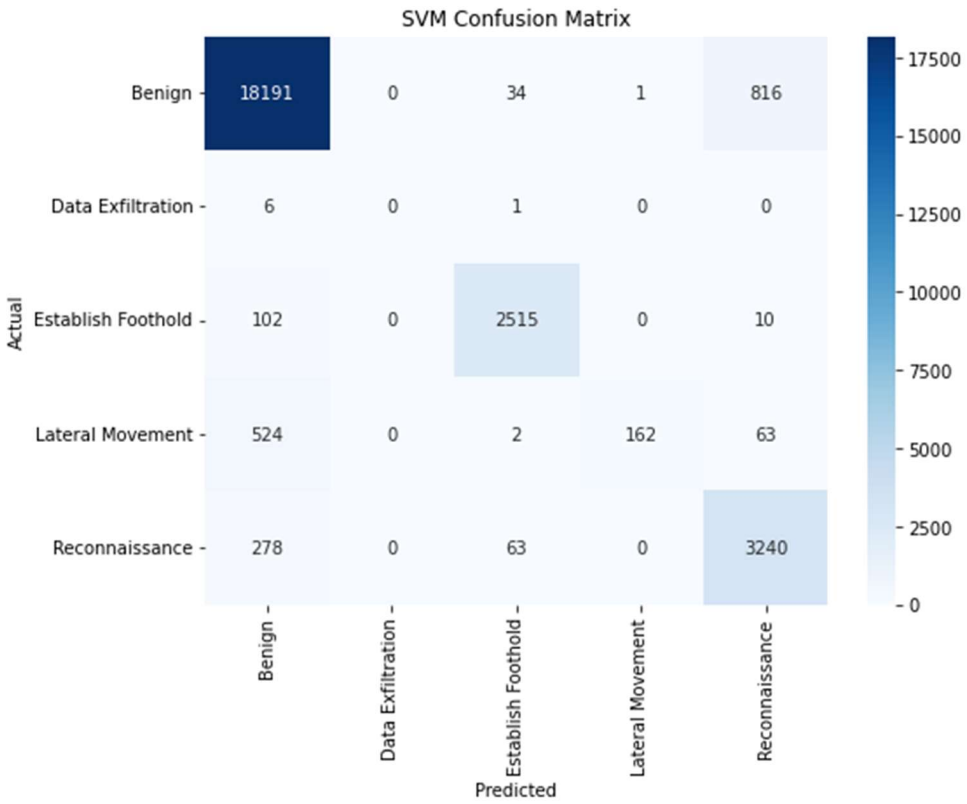


Fig. 8: Confusion matrix for SVM

The following figure shows a graphical representation of the correlation coefficient for the selected features in the form of a heat map, where it can be seen that several of them are mutually interlinked.



Fig. 9: Heat map of the correlation coefficient for the selected features

The following figure shows the 10 most relevant features for APT attack detection in the random forest model, evaluated on an impurity basis (impurity-based).

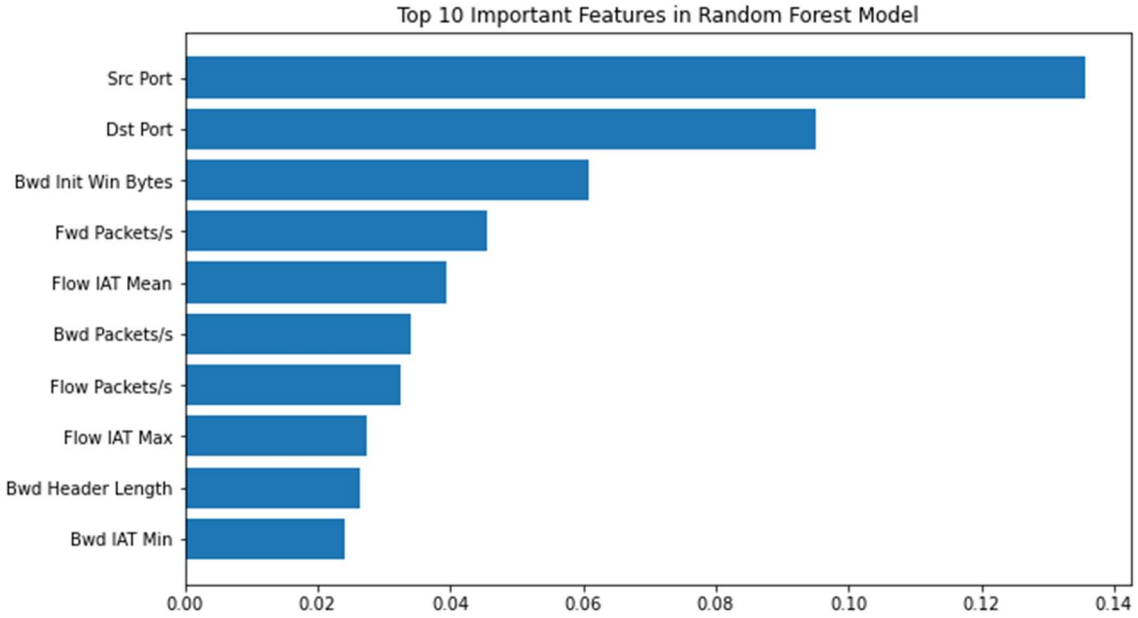


Fig.10: The 10 most important features in the random forest model

The last figure visualizes the success rate (the Accuracy parameter) of each of the selected classification methods.

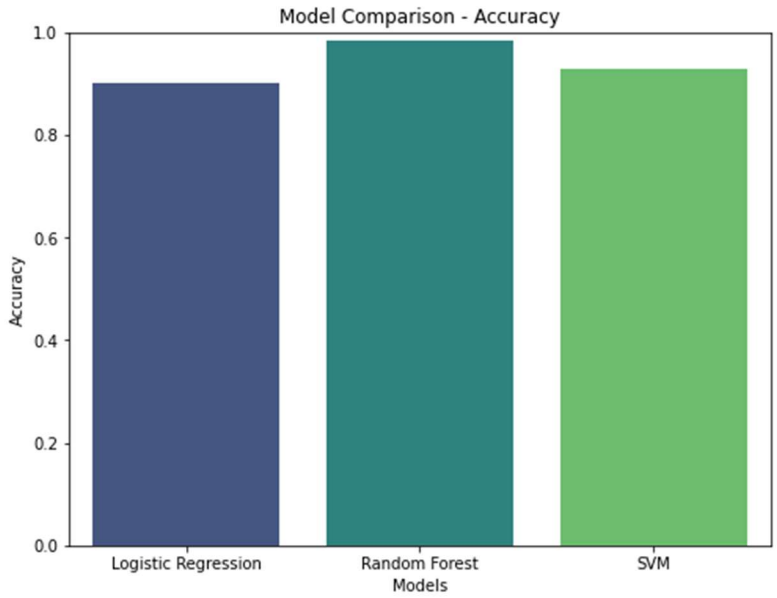


Fig. 11: Comparison of model success rates

Summary

Based on the experiments performed, it can be concluded that the most suitable method for detecting advanced APT attacks is the most suitable method, the *Random forest* method. On the DAPT2020 dataset, which is one of the most comprehensive and most frequently used datasets for APT attack detection, we achieved a detection success rate of 98%. This value is almost identical for both the Precision and Recall metrics, so the detection of both false positive and false negative detections is very well covered. The second most successful method, with a detection success rate of 92%, is the *SVM* method, while the *logistic regression* method achieves a comparable detection success rate of ~90%. In terms of the time needed to test the data, the *Random forest* and *logistic regression* methods achieve very comparable results in the order of tens of milliseconds, so both methods can be deployed for real-time applications. The *SVM* method requires a significantly longer detection time during testing, in the order of tens of seconds, which represents more than 100 times longer.

Based on the experiments performed, it can therefore be summarized that the most suitable method for detecting advanced APT attacks is the most suitable method **Random Forest.**