PROJECT

Multi-tenant cybersecurity operations centre designed
as an open cloud service with machine learning elements

V17 Test results and evaluation

KPB5 – Implementation of the solution

type – report

month of submission – M24

Funded by the European Union
NextGenerationEU

RECOVERY AND RESILIENCE PLAN

GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

Contents

1 Machine learning and artificial intelligence in cybersecurity 4
1.1 Detection of unwanted e-mail messages (Spam) 4
Evaluation of tests 5
Test results 7
Assessment of tests 8
1.2 Detection of DoS/DDoS attacks 8
DoS (Denial of Service) 8
DDoS (Distributed Denial of Service) 9
Analysis of statistical parameters on selected DDoS attacks 10
Variability, skewness and kurtosis 12
Detection values for autocorrelation and regression 15
Response of divergence to a DDoS attack 17
Use of the 3σ-Tunnel for recognizing DDoS attacks 17
Detection functions for machine recognition of DDoS attacks 19
Assessment of tests 21
1.3 Detection of malicious code (Malware) 22
Preparation and testing procedure 23
Test results 23
Assessment of tests 24
1.4 Phishing detection 25
Preparation and testing procedure 25
Test results 25
Assessment of tests 30
1.5 Detection of advanced persistent threats (APT) 31
Evaluation of tests 33
Test results 33
Assessment of tests 36
2 Component tests in the INNOSOC PoC 37
2.1 SIEM - ELK Stack 37
2.2 Shuffle 37
2.3 Wazuh 37
2.4 Greenbone Community edition 37
2.5 OpenCTI 37
2.6 MISP 37
2.7 GLPI 37

Funded by the European Union NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

2.8 Netbox 37
2.8.1 TS-NB-01 Verification of the availability and basic functionality of NetBox 37
2.8.2 TS-NB-02 Verification of the recording of infrastructure objects 39
2.8.3 TS-NB-03 Verification of relationships and orientation in the infrastructure context 40
2.8.4 TS-NB-04 Verification of the use of NetBox as a source of context for the PoC 41
2.8.5 Assessment of tests.....41
2.9 DFIR IRIS 41
2.10 Kibana (ELK) 41

1 MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE IN CYBERSECURITY

With the advent of the information age, cybersecurity has become a pressing issue in today's society and a skill that is in high demand in industry. Businesses, governments and individual users are all exposed to the risk of security attacks and breaches. The fundamental goal of cybersecurity is to protect users and their data. Cybersecurity is a multifaceted problem, ranging from highly technical areas (cryptography and network attacks) to user-facing areas (detection of hate speech or fraudulent credit card transactions). It helps prevent damage to sensitive information, prevent financial fraud and losses, and protect users and their devices from malicious actors. A large part of cybersecurity analytical processes, investigation and detection is currently driven by machine learning ML (Machine Learning) and "intelligent" systems.

In the following text, we describe the test results for the following defined security threats:

- spam
- DoS/DDoS attacks
- malware
- phishing
- advanced persistent threats APT

1.1 Detection of unwanted e-mail messages (Spam)

Spam e-mails (unwanted messages) account for approximately 60% of global e-mail traffic. Although spam detection software has improved significantly since the first spam message in 1978, anyone who has an e-mail account knows that spam continues to be a time-consuming and costly problem.

From the wider range of commonly used methods and algorithms for detecting and distinguishing between harmless/solicited e-mail messages (ham) and unwanted/unsolicited messages (spam), we selected and tested the following methods:

- Decision Tree
- SVM
- XGBoost
- Naive Bayes
- Logistic regression

In an effort to create a sufficiently robust and universal system, we used the following datasets for our experiments:

- SpamAssassin
- SMS Spam

The *SpamAssassin* dataset (also called the SpamAssassin Public Corpus) is a collection of e-mail messages that have been labelled as spam or ham. The database was created by the Apache SpamAssassin project, which is an open-source anti-spam platform. The corpus contains several thousand e-mail messages divided into the spam and ham categories. In total, the dataset contains 6,047 messages with a spam ratio of approximately 31%. Spam messages represent unsolicited commercial e-mails, phishing attempts and other

unwanted communication, while ham messages represent legitimate personal and professional e-mails. The e-mail messages are stored as raw text files in the standard e-mail format (RFC 822), preserving headers, body content and metadata. The dataset contains various types of e-mails with different formatting styles, HTML content, references to attachments and various languages (although English predominates). In terms of content, these are real e-mails collected from actual users who donated their messages, which makes them representative of real spam and ham patterns. The dataset includes complete e-mail headers, which contain valuable metadata such as sender information, routing details, timestamps and content type information. In the SpamAssassin corpus, the individual samples are grouped into several categories:

- easy_ham – legitimate e-mail messages that can easily be distinguished from spam, as they usually do not have any "spam" signatures, such as an excessive amount of HTML code
- hard_ham – messages that are not spam but which in many respects resemble typical spam, which usually uses HTML code, unusual HTML tags, coloured text or "spammy-sounding" phrases
- spam – unwanted spam e-mail messages

This dataset is most often used for training and evaluating spam classification algorithms, comparing machine learning models for text classification, research in natural language processing and information retrieval, or testing feature extraction techniques for e-mail filtering. Over the years, the dataset has been released in several versions, and the public SpamAssassin corpus is freely available for research purposes.

The *SMS Spam* dataset is a freely available database of SMS messages collected for the purposes of research into unwanted spam messages from mobile phones. The dataset was compiled from several sources and contains 5,574 English, real and unencoded messages, labelled as legitimate messages of the ham type or unwanted messages of the spam type.

Evaluation of tests

The following metrics were used to evaluate the experiments.

1.1.1.1 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. 1: Confusion matrix

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

- True Positive (TP) – Number of correct 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 incorrect positives: This is the number of total cases from the negative class which 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 out 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 incorrect negatives: This is the number of the total number of cases from the positive class which our model incorrectly classified by predicting them as negative. Hence the name, false negative.

1.1.1.2 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 its calculation from the confusion matrix is as follows:

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

1.1.1.3 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.

1.1.1.4 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.

1.1.1.5 F1-Score

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

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

Test results

SpamAssassin

Method	Accuracy	Precision	Recall	F1-score	Processing time (sec.)
Decision tree	0.9761	0. 9776	0.9774	0.9774	1.4
SVM	0.9928	0.9929	0.9928	0.9929	11.9
XGBoost	**0.9928**	**0.9929**	**0.9928**	**0.9929**	**3.3**
Log. Regression	0.9833	0.9833	0.9833	0.9833	1.2
Naive Bayes	0.8438	0.8624	0.8439	0.8330	1.3

Tab. 1: Test results for the SpamAssassin dataset

SMS Spam

Method	Accuracy	Precision	Recall	F1-score	Processing time (sec.)
Decision tree	0.7585	0.7624	0.7585	0.7604	0.001
SVM	0.7706	0.7722	0.7706	0.7714	0.005
XGBoost	0.7826	0.7757	0.7826	0.7791	0.001

Log. Regression	**0.7887**	**0.7709**	**0.7888**	**0.7795**	**0.001**
Naive Bayes	0.7785	0.7733	0.7785	0.7759	0.001

Tab. 2: Test results for the SMS Spam dataset

Evaluation of the tests

Based on the experiments carried out, it can be concluded that the most suitable method for detecting spam e-mail messages in terms of effectiveness is the XGBoost method. For the "SpamAssassin" dataset, it achieves a spam detection success rate of 99%. This value is almost identical for both the Precision and Recall metrics, so the detection of both false positive and false negative detections of spam messages is very well covered. For this dataset, the SVM method also achieves almost identical results; however, the evaluation time for the SVM algorithm is approximately 4x longer than for XGBoost. Nevertheless, it is still a process of a few seconds, so both methods can be deployed for real-time applications. For the "SMS Spam" dataset, the spam detection success rate dropped significantly to ~78%. The best results were achieved by the Logistic regression method, although the difference between the XGBoost and Logistic regression methods is only tenths of a percent. In terms of the time required for data prediction, the SVM algorithm again requires the longest interval. Based on the experiments carried out, it can therefore be summarized that the most suitable method for detecting spam messages is the **XGBoost** algorithm.

1.2 Detection of DoS/DDoS attacks

"Denial-of-service" (DoS) and "distributed denial-of-service" (DDoS) attacks aim to disrupt the normal operation of a target server, service or network by overwhelming it with a flood of internet traffic. DoS attacks usually originate from a single source, while DDoS attacks are launched from multiple compromised devices, often dispersed globally. This multitude of sources makes DDoS attacks harder to stop, and they can have a more significant impact.

These attacks work by overloading the target with more requests than it can handle, causing the system to slow down significantly or crash completely and denying service to legitimate users. Common methods include exploiting server vulnerabilities or using a network of bots, known as a botnet, to flood the target with traffic. The motivations for these attacks can range from simple vandalism and personal grudges to extortion and political activism.

DoS (Denial of Service)

The abbreviation DoS translates as an attack denying a service. This type of attack is used to overwhelm a server. In this case, the attacker uses a device with which they attempt to disrupt the normal operation of the target server, service or network. They very quickly became a cybersecurity threat. They are simple, but their impact can be devastating for individuals as well as companies.

DoS attacks are often used to shut down devices and networks, and there are many different ways of using the above DoS attacks. These include, for example:

- Buffer overflow: A buffer is temporary storage for storing data. If a program or system process places more data than was predetermined, the additional data then overflow and some of them may then leak into other buffers. This then results in the corruption or overwriting of other data. Based on this principle, an attacker can corrupt files and alter or disclose various information.
- ICMP attack (ICMP attack or Ping of Death): This type of attack uses the ICMP protocol, employing ICMP echo requests, so-called pings. In general, echo

requests and echo replies are used to ping a network device in order to verify the connectivity between the device and the sender. If an attacker flooded the target with request packets, the network must respond with the same number of packets, which makes the device unavailable for other incoming requests.

- UDP attack (UDP attack): The attack is based on sending UDP packets with a source IP address that the attacker can spoof, the aim being to flood random ports on the target server. The advantage of the UDP protocol is that it does not need to establish a connection, as is the case with the TCP protocol. If the target server receives a UDP packet on a port that is not used by any application, it informs the sender about this. If the attacker sends a sufficient number of packets to such a port, it may become overloaded.
- SYN attack (SYN flood): This is an attack that arises when a TCP connection is established between 2 devices. TCP uses a three-way handshake. Establishing a connection begins with sending a SYN message. If the server is active, a SYN/ACK message is sent back to the client that wanted to establish the connection. The server tries to respond until the time for establishing communication expires, while keeping the data in memory. If the client is an attacker with a spoofed and unreachable source address, sending a large number of such packets can exhaust the memory allocated for TCP connections, and all further communication attempts will then be ignored.
- Teardrop attack (Teardrop attack): During this attack, the attacker tries to send fragmented packets, and the network then tries to compile these data back into their original state. The attacker can exploit a vulnerability of an outdated operating system. It is common for many companies to still use older versions of systems in order to keep running an old application, and since some versions of operating systems, such as Linux or Windows, contain a bug in reassembling a fragmented packet, the server may be attacked precisely because of this vulnerability.

DDoS (Distributed Denial of Service)

Unlike a simple DoS attack, a DDoS (Distributed Denial of Service) attack gains strength because the attacker uses multiple systems at once, thereby increasing the severity of the targeted attacks. The attacker may use malware that they send over the network to devices with the intention of infecting them. They thus become part of a so-called botnet and, following the attacker's commands via a command-and-control server, these bots then attack the targeted server in unison. DDoS attacks are very difficult to detect because the attacking bots may be located in different places around the world. The main and key difference between a DoS and a DDoS attack is that with DDoS, using bots, the attacker can send huge volumes of traffic.

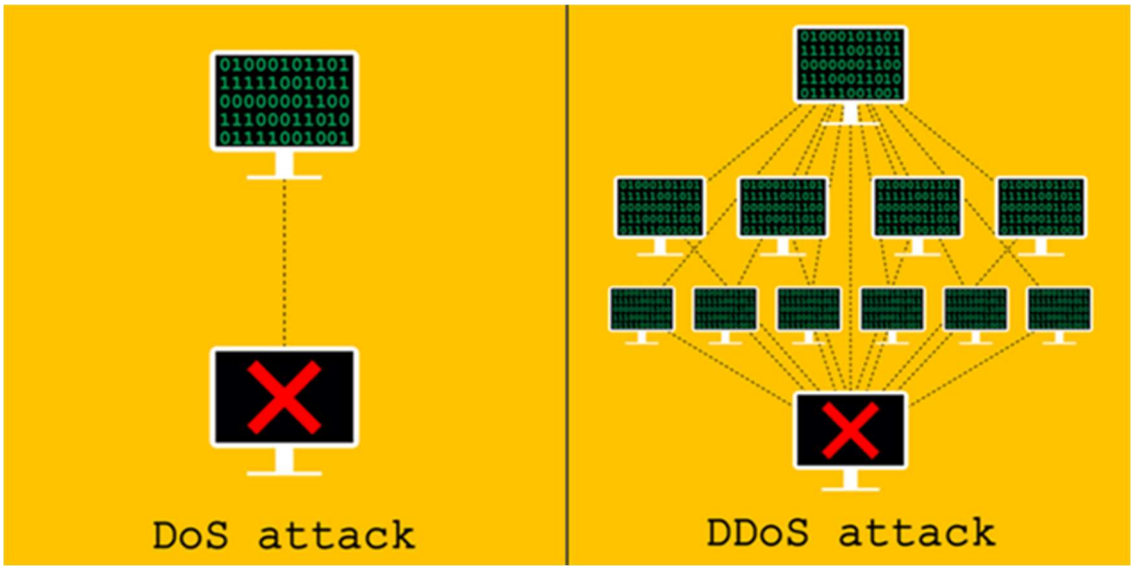


Fig. 2: DoS and DDos attacks

DDoS attacks are divided into 3 main categories:

1. Volumetric attack: This type exploits vulnerable services such as NTP, DNS or SSDP. The attacker tries to flood the target with a large volume of packets to which the target tries to respond, until the targeted network infrastructure collapses. It is usually a broadband attack that can exceed 100 GB or even several TB per second and is immediately obvious to connectivity providers. Network monitoring, flow analysis and flow telemetry analysis have proved effective for detecting volumetric attacks.
2. Protocol attack: This type of attack tries to exhaust the resources of the server or its systems and relies on vulnerabilities of internet communication protocols. Protocol attacks include, for example, the SYN attack, UDP attack and ICMP attack. These attacks are considered sufficiently crippling for the targeted infrastructure owing to the frequency of their duration. In 2018, attackers used the BGP protocol to redirect traffic intended for the “MyEtherWallet” service, which, however, was merely a cover for the theft of cryptocurrency wallets. It lasted approximately 2 hours.
3. Application layer attack: This attack targets a specific application and its vulnerabilities so that it is unable to process, send or receive data. The attacks are weak and slow and concentrate on GET and POST requests. The aim is then to cripple the web server, and they are measured in requests per second (Rps). They are low and weak so that they comply with the protocol.

There are several symptoms that can help us detect whether our network, application or the entire infrastructure is under a DDoS attack. This may be a sudden increase in web activity on the server, reduced network performance or a sudden inability to communicate with the application.

Analysis of statistical parameters on selected DDoS attacks

We will present the response of selected statistical parameters on various records of DDoS attacks. We have omitted a number of experiments that we carried out on various simulated scenarios. On these scenarios, the selected statistical parameters worked “as in a textbook”, and we gained an initial idea of the effectiveness of using the individual parameters to recognize a change and an increase in the simulated traffic. When deployed to recognize

a real attack, the responses of the statistical parameters were no longer so unambiguous. For the presentation, we selected eight different records of DDoS attacks.

The first four records represent standard DDoS attacks (N – normal), in which the parameters behave similarly to the simulated data. The next two records also represent standard attacks at first sight, but the parameters behave in a substantially different, indeterminate manner in them (S – special). The seventh record is a representative of so-called Low Rate (LR) DDoS attacks, i.e. attacks with a low average flow intensity. The last record is an attack which, in our opinion, cannot be detected by the above statistical parameters (P – as in problem). For the records, together with the number of packet increments a(t), we also displayed their moving average m(t) with a compute window cw = 1024 ts.

The records are sampled into given time windows (ts – time slot), mostly ts = 1ms. For the step-by-step calculation of the parameters, we used a shift of the so-called compute window (cw) by one ts. When conducting the experiments, we used various powers of two (256,512,1024,2048,4096) for the compute window sizes because of the estimation of the Hurst exponent. After a subjective evaluation of the course of the values of the parameters examined, we decided to focus further mainly on a compute window of size cw = 1024 ts. With cw = 512 ts, significant "undulation" occurred in some parameters; with cw = 2048, the time needed to calculate the parameters, especially the Hurst exponent, increased significantly.

In the graphs of the DDoS attack records, we have also displayed the course of the moving average and shown the time interval in which the moving average m(t) reached its local maximum from the start of the attack. We will use this interval in the subjective evaluation of the other statistical coefficients. In all records of DDoS attacks, we can observe a linear growth of the moving average. We are looking for coefficients whose values start to grow faster than linearly at the start of the attack.

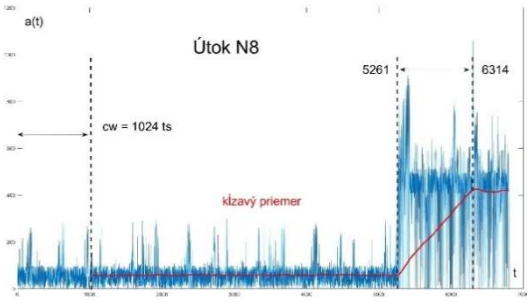


Fig. 3: Attack N8 and moving average

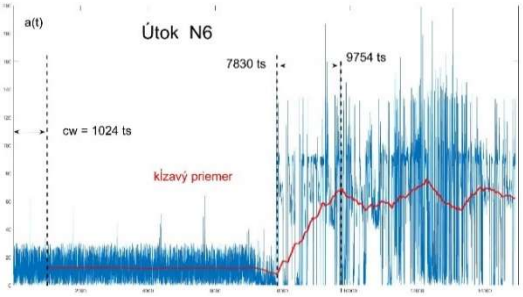


Fig. 4: Attack N6 and moving average

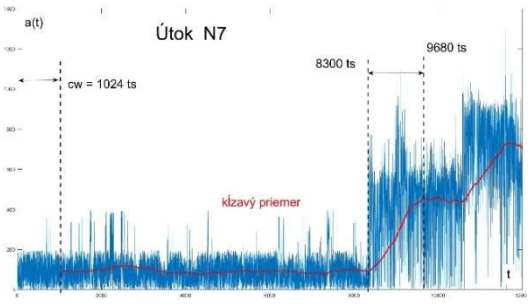


Fig. 5: Attack N7 and moving average

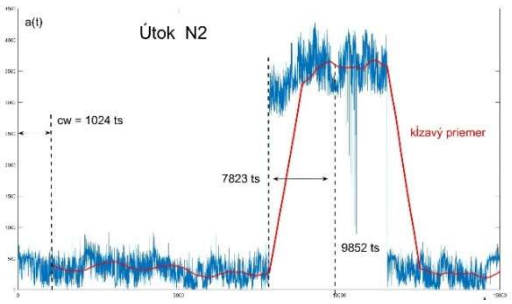


Fig. 6: Attack N2 and moving average

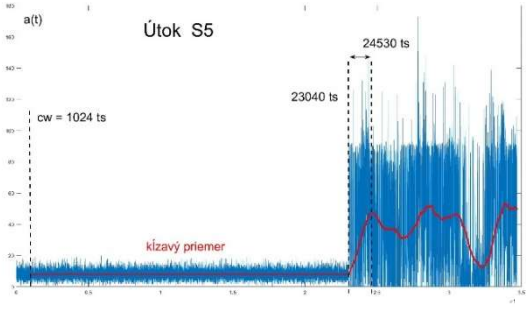


Fig. 7: Attack S5 and moving average

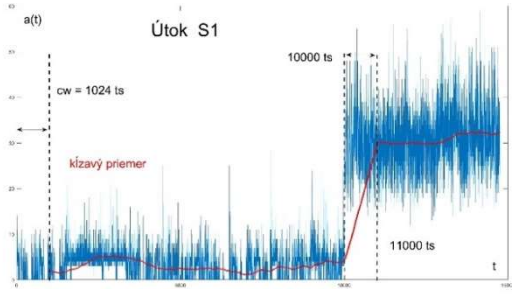


Fig. 8: Attack S1 and moving average

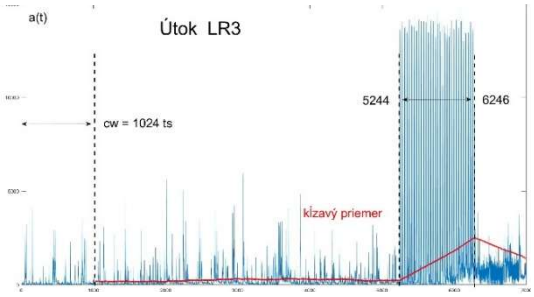


Fig. 9: Attack LR3 and moving average

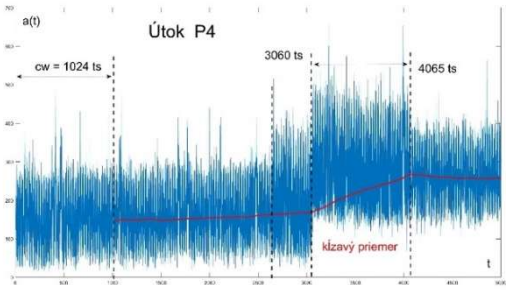


Fig. 10: Attack P4 and moving average

Variability, skewness and kurtosis

First, we will deal with the basic statistical moments, or rather the coefficients of variation V(t), skewness K(t) and kurtosis Skw(t). The coefficient of variation is the ratio of the standard deviation to the mean value of the increments in the given time window cw. Variability expresses the degree of dispersion of the increments a(i) around the mean value. In the first graph we show the course of the moving average and the standard deviation; the second graph shows the values of the coefficient of variation:

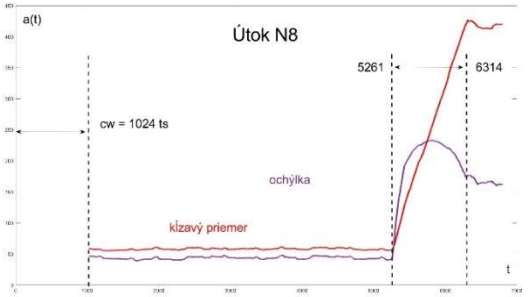


Fig. 11: Attack N8, mean and deviation

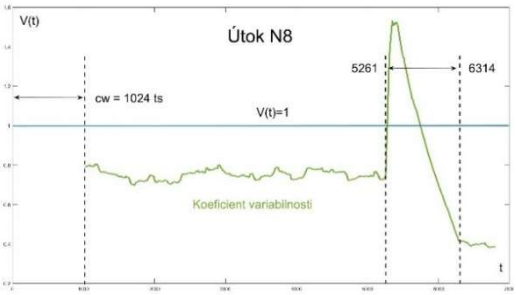


Fig. 12: Attack N8, coefficient of variation

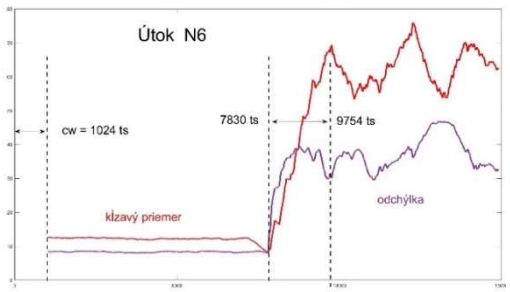


Fig. 13: Attack N6, mean and deviation

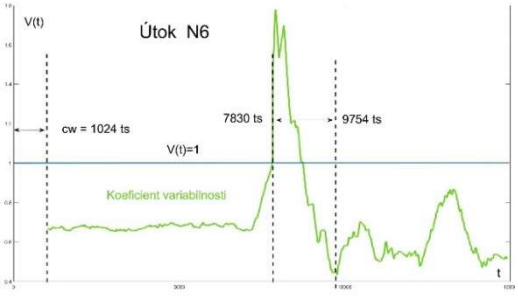


Fig. 14: Attack N6, coefficient of variation

In the case of the standard DDoS attacks (N2,N6,N7,N8), the coefficient of variation responded in the same way. Standard traffic represented a stationary flow with a deviation smaller than the mean flow intensity. At the moment of the onset of the attack, the deviation responded with a faster increase in values than the linear growth of the moving average; therefore, the ratio of the deviation to the mean reached the value V(tA) = 1 at the moment of the attack. This fact could be used for fast and simple detection of this type of attack. However, this is not a rule; in Fig. 15, for attack S1, we can see that stationarity is disrupted during standard traffic and the coefficient exceeded the value V(tA) = 1 several times before the attack.

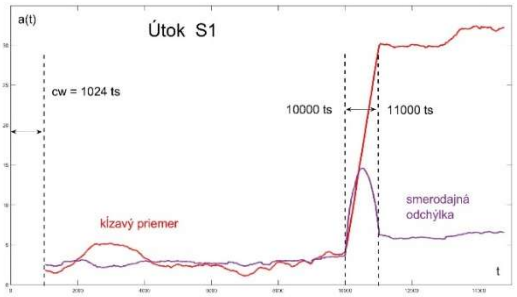


Fig. 15: Attack S1, mean and deviation

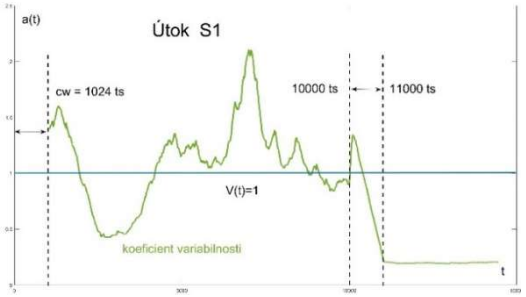


Fig. 16: Attack S1, coefficient of variation

Figs. 17 and 18 show the graphs of the mean, deviation and coefficient of variation for the so-called Low Rate DDoS attack LR3.

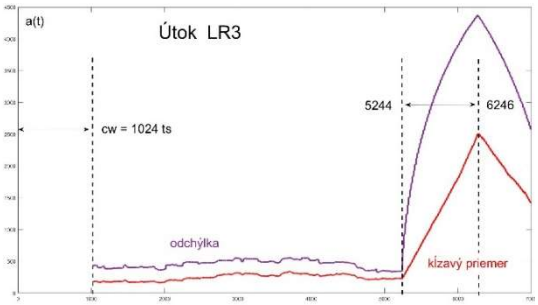


Fig. 17: Attack LR3, mean and deviation

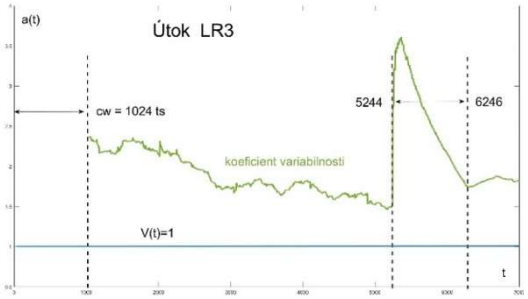


Fig. 18: Attack LR3, coefficient of variation

In the case of LR3, the deviation during normal traffic is greater than the mean flow intensity, so the coefficient of variation is V(t)>1 throughout. The value 1 cannot now be used for fast detection. However, at the moment of the attack, the deviation values again increased faster than the linear growth of the mean intensity, which manifested itself in a marked step increase in the coefficient of variation. It should be possible to recognize this change using prediction methods.

Other simple statistical parameters are the coefficient of skewness and the coefficient of kurtosis. These are in fact the third and fourth central moments scaled by the standard deviation.

$$\mu_3 = \frac{1}{\sigma^3} E[X - EX]^3 \qquad \mu_4 = \frac{1}{\sigma^4} E[X - EX]^4 \qquad \sigma^2 = E[X - EX]^2$$

Funded by the European Union
NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

In our case, the realizations of the variable X are the values of the increments a(t), and the coefficients describe the properties of the probability distribution of the increments in the given compute window cw. As the attack traffic was progressively loaded into the evaluated window cw, we assumed a marked change in the probability distribution and consequently also a change in the coefficients. This assumption was confirmed for almost all the flows analysed. Both statistical coefficients responded with a marked increase in their values right at the start of the DDoS attack.

First, we present graphs of the coefficients applied to two selected standard attacks, followed by some problematic situations.

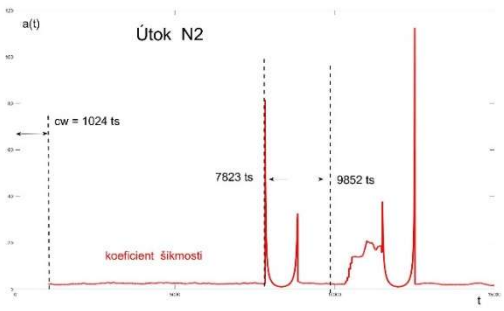


Fig. 19: Attack N2, coefficient of skewnes

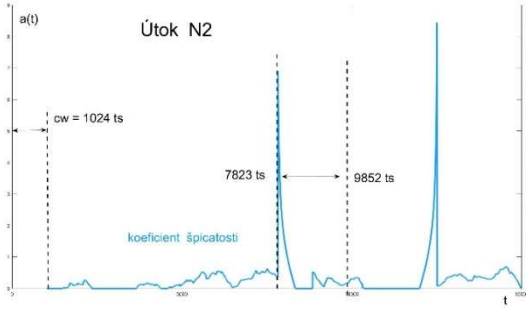


Fig. 20: Attack N2, coefficient of kurtosis

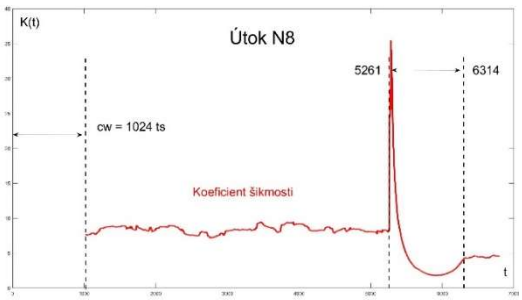


Fig. 21: Attack N8, coefficient of skewne

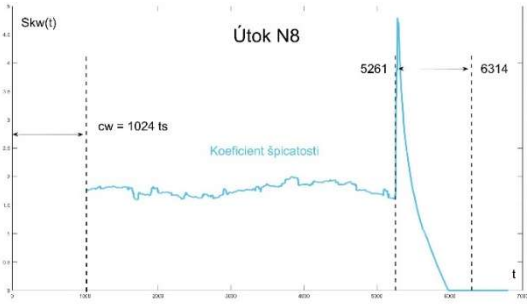


Fig. 22: Attack N8, coefficient of kurtosis

In standard type N attacks, the coefficients of skewness and kurtosis fluctuate only slightly during calculation on standard traffic owing to stationarity, and at the moment of the onset of the DDoS attack their values increase markedly. We assume that we will be able to recognize this marked jump automatically using suitable prediction methods.

A considerably less favourable situation arose with non-standard attacks, e.g. S1. Owing to the non-stationarity of normal traffic and the frequent occurrence of high peaks, both coefficients oscillate strongly, as a result of which the jump at the onset of the attack is lost in the overall course of the values of both coefficients. In such a case, the onset of the DDoS attack cannot be recognized by machine.

To visualize the response of the coefficients to a change in the character of normal traffic and to the occurrence of peaks, we displayed the values of the coefficients in a common graph together with the increments of flow S1.

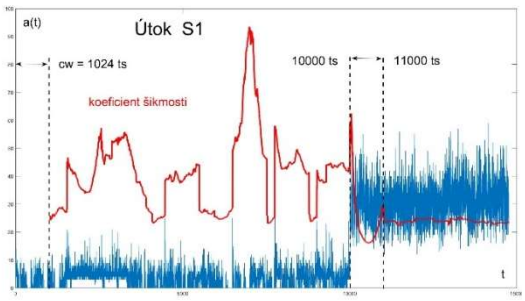


Fig. 23: Attack S1, coefficient of skewness

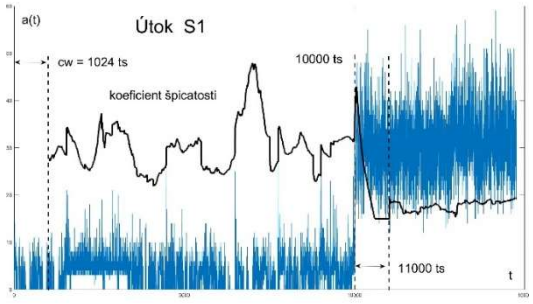


Fig. 24: Attack S1, coefficient of kurtosis

The response of the coefficients to the attack with a low mean intensity LR3 turned out as positively as for the standard N-type attacks.

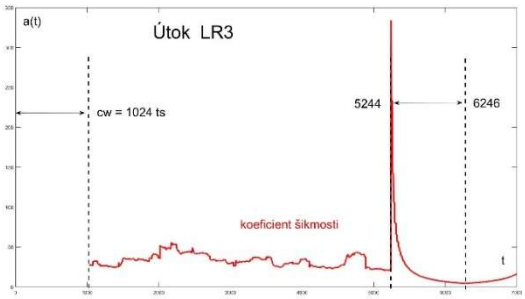


Fig. 25: Attack LR3, coefficient of skewnes:

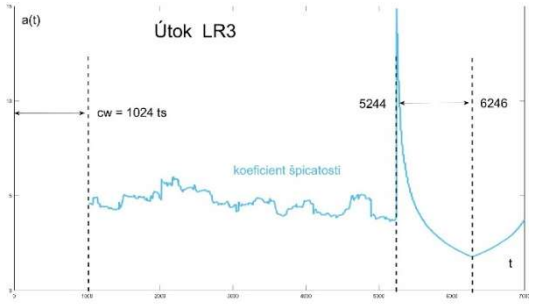


Fig. 26: Attack LR3, coefficient of kurtosis

Again, we can see the possibility of automatically distinguishing the marked increase in the values of the coefficients at the start of the attack from the values describing normal traffic.

Detection values for autocorrelation and regression

The next two coefficients describe the autoregressive and correlation dependence between two consecutive cw windows, or rather between the random variables X(t) and X(t-1), whose realizations are the flow increments a(t) in the windows considered.

$$X(t) = c(t).X(t-1) + \varepsilon(t), \qquad \rho(t) = \frac{E[(X(t) - EX(t))(X(t-1) - EX(t-1))]}{\sigma(t)\sigma(t-1)}$$

A great advantage of these coefficients is that their values range within the interval (-1,1). At the moment the attack was loaded into the evaluated cw windows, for most attacks their values began to rise rapidly towards the upper limit of 1. For this reason, it is possible to set a certain limit value (threshold) for both coefficients, the exceeding of which would signal the beginning of a DDoS attack. The lower this value, the earlier the attack is signalled; on the other hand, attacks that are too low may cause false alarms. Based on the experiments carried out, and in an effort to identify as many DDoS attacks as possible without false alarms, we set the limit for the autoregressive coefficient to c(t) = 0.9 and for the correlation coefficient to ρ(t) = 0.75. In the following graphs, we again show the evolution of the coefficients for attacks N8, N2, S1 and LR3. Attack N8 represents the behaviour of the coefficients in most standard attacks; in N2 we see a standard attack in which, however, the autocorrelation coefficient has all values greater than 0.9 and the detection cannot be used. In the non-standard attack S1,

both coefficients exceeded the detection levels during the attack. For the low-intensity attack LR3, this method is unusable.

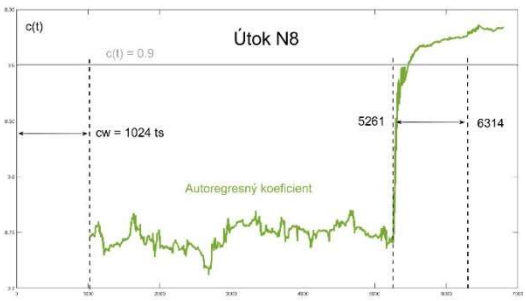


Fig. 27: Attack N8, Autoregressive coefficien

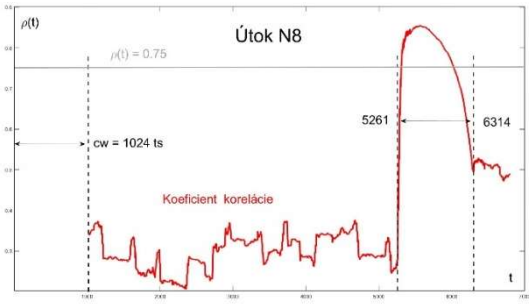


Fig. 28: Attack N8, Correlation coefficient

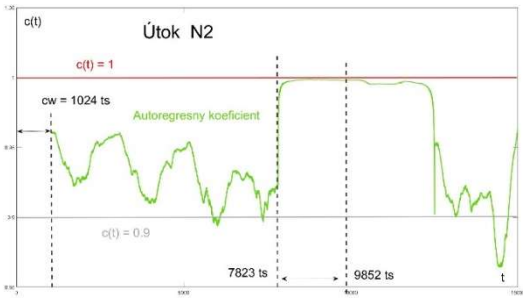


Fig. 29: Attack N2, Autoregressive coefficier



Fig. 30: Attack N2, Correlation coefficient

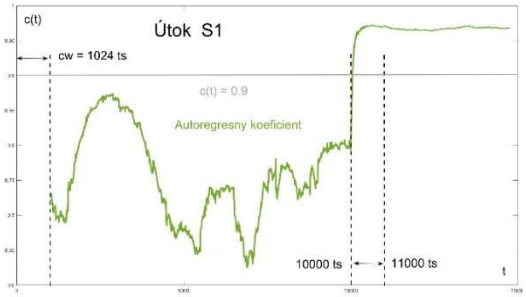


Fig. 31: Attack S1, Autoregressive coefficien

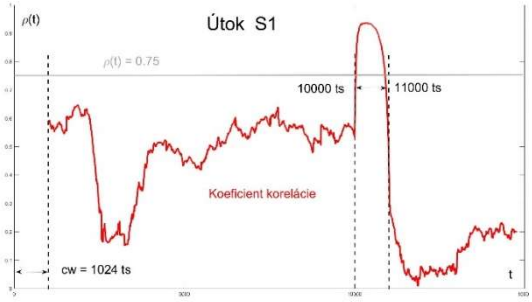


Fig. 32: Attack S1, Correlation coefficient



Fig. 33: Attack LR3, Autoregressive coefficie

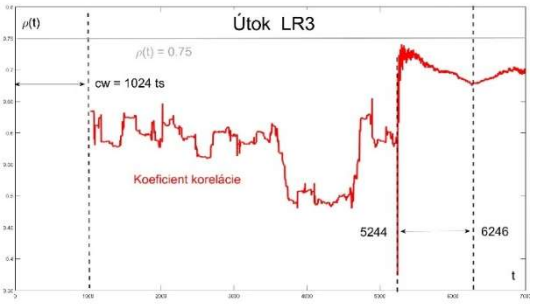


Fig. 34: Attack LR3, Correlation coefficient

For most attacks, it can be assumed that exceeding the detection values for autoregression and correlation can be used, in combination with other coefficients, for early and simple detection of a flood attack.

Response of divergence to a DDoS attack

The last analysed parameter, the Kullback-Leibler divergence, in a certain way compares the probability distribution P(.) of the increments a(t) in successive overlapping computation windows cw.

$$dvg(t) = \sum_{k=1}^{n} P(t-1) ln(\mathrm{P(t-1)/P(t)})$$

The advantage of divergence is that, on the samples monitored, in most cases it responded immediately to the onset of a DDoS attack with a high impulse (peak). The disadvantage is that it tended to respond in this way to peaks in normal traffic as well, which could cause a large number of false alerts.

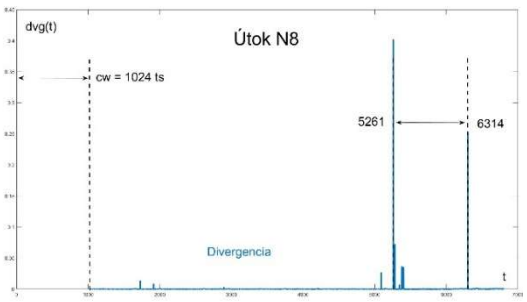


Fig. 35: Attack N8, Divergence

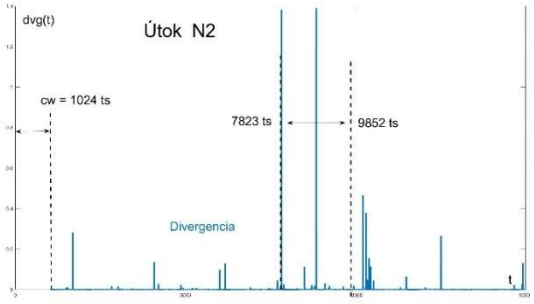


Fig. 36: Attack N2, Divergence

A further disadvantage of using divergence is the fact that the high impulse at the onset of an attack lasts a very short time compared with the other parameters; we therefore consider its application in combination with the other parameters to be problematic. We have therefore postponed the use of divergence to the future.

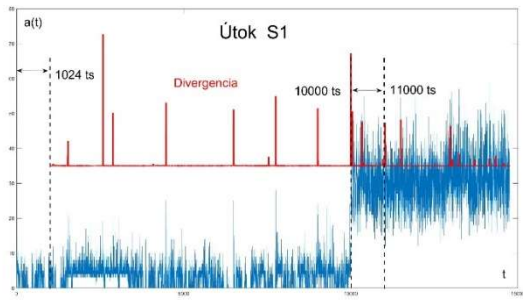


Fig. 37 Attack S1, Divergence

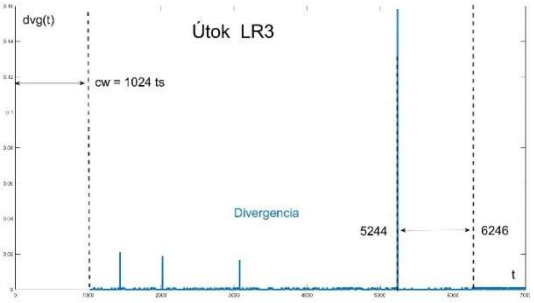


Fig. 38: Attack LR3, Divergence

Use of the 3σ-Tunnel for recognizing DDoS attacks

Based on the experiments, we chose 1000 previous values for calculating the mean and deviation, and we chose the interval width (EX-3σ, EX+3σ) (with a smaller interval size, frequent false alarms occurred). We called this method the predictive 3σ-tunnel. A natural question arises as to why not apply the predictive tunnel directly to the IP flow increments a(t) and not bother with calculating statistical coefficients. After carrying out experiments, we found that although exceeding the upper limit of the tunnel detects the onset of a DDoS attack

relatively early, false alarms are also frequently reported. When the multiple of the standard deviation was increased, the number of false alarms decreased, but at the same time the ability to detect the attack decreased. In the following figures, we will show only the upper limit of the 3σ-Tunnel for selected attacks.

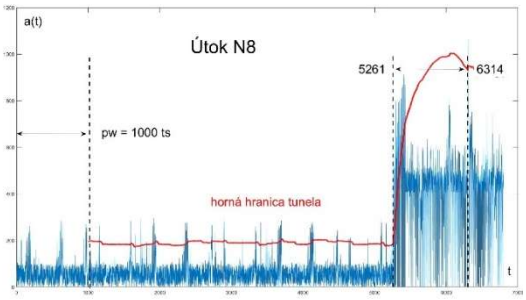


Fig. 39: Attack N8, upper limit of the 3σ-Tun

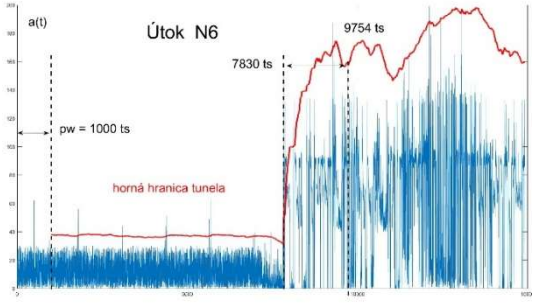


Fig. 40: Attack N6, upper limit of the 3σ-Tunnel

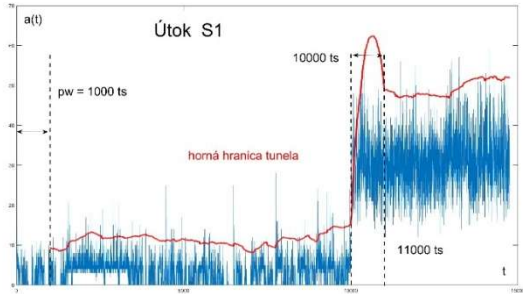


Fig. 41: Attack S1, upper limit of the 3σ-Tunr

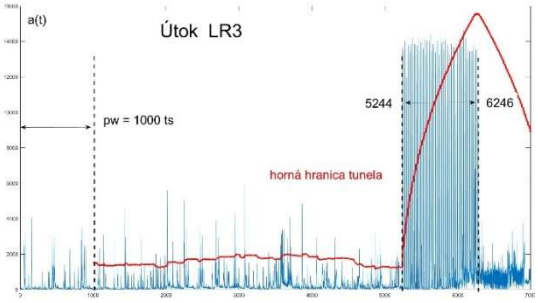


Fig. 42: Attack LR3, upper limit of the 3σ-Tunnel

In the following graphs, we will demonstrate the behaviour of the 3σ tunnel on the parameter with the greatest variability among the statistical coefficients considered, namely the Hurst exponent (estimated by RS statistics with removal of AR1 autoregression).

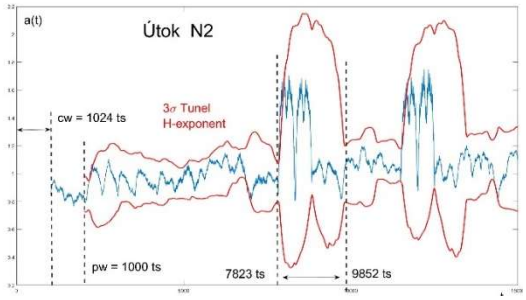


Fig. 43: N2, 3σ-Tunnel for the H-exponent

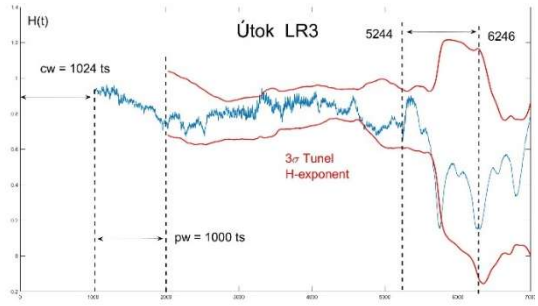


Fig. 44: LR3, 3σ-Tunnel for the H-exponent

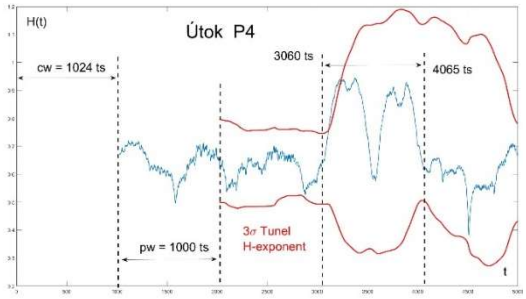


Fig. 45: P4, 3σ-Tunnel for the H-exponent

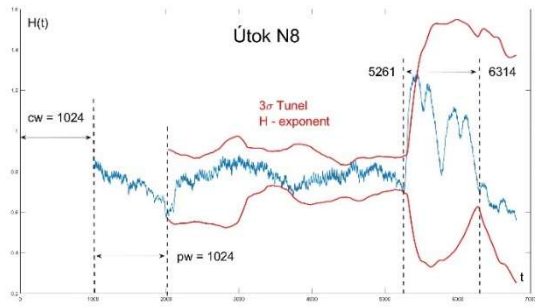


Fig. 46: N8, 3σ-Tunnel for the H-exponent

In most cases (except for attack LR3), the 3σ tunnel recognized the start of the DDoS attack early, with a minimal, sometimes even zero, number of false alerts.

As we have already stated, on the basis of experiments we set the size of the prediction window to 1000 ts and the interval width for the 1001st value to three times the standard deviation. With this setting, in most experiments (various coefficients and various attacks) we obtained early recognition of the start of the attack with a minimal number of false alerts. In the preceding figures, this is 3 cases out of 4. A tunnel set up in this way has two useful properties:

- It has a sufficiently “long memory” to be able to cover the local variability of the statistical coefficient and not cause a large number of false alerts
- It has a sufficiently "long memory" so as not to be able to respond to a relatively large increase in the values of the investigated coefficient at the start of a DDoS attack (large relative to the relative variability)

Of course, in order to be able to draw more relevant conclusions and recommendations, it is necessary to carry out a large number of experiments on records of various DDoS attacks.

Detection functions for automatic recognition of DDoS attacks

In the preceding chapters, we presented two ways of recognizing DDoS attacks using statistical coefficients. The first way was to determine detection values for suitable parameters: 1.00 for variability V(t) and the H-exponent, 0.90 for the autoregressive coefficient c(t) and 0.75 for the correlation coefficient ρ(t). The second way, using the predictive tunnel, was analysed in the preceding section.

When using the limit values, we introduced a two-valued logical function Y[.], which evaluates the exceeding of the limit value for a given statistical coefficient. In the following graphs, we will show the variability V(t), the autoregressive coefficient c(t) and the correlation coefficient ρ(t) and the course of their function Y[.] for attacks S1a N2.

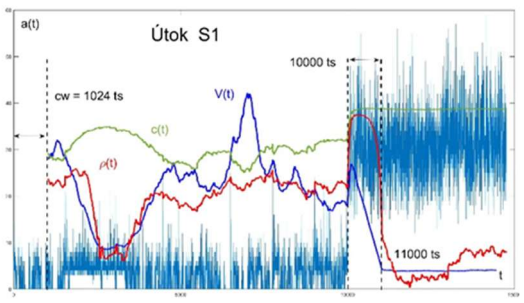


Fig. 47: S1, coefficients V(t), c(t), ρ(t)

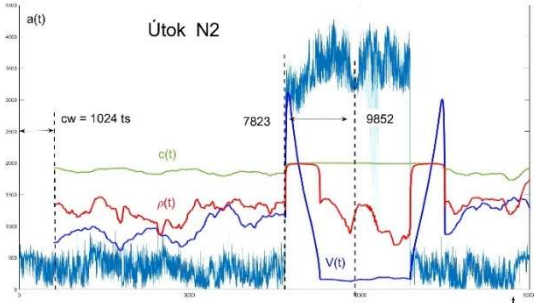


Fig. 48: N2, coefficients V(t), c(t), ρ(t)

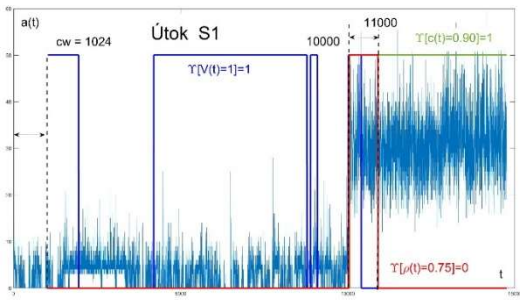


Fig. 49: S1, course of the functions Y[.]

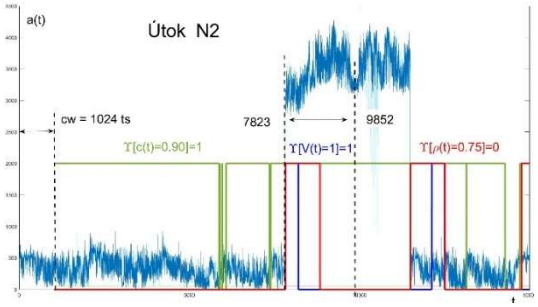


Fig. 50: N2, course of the functions Y[.]

The sets of values on which the coefficients exceed their detection limits will be called detection intervals. In Figs. 49 and 50, we can see how these intervals overlap one another. In order to eliminate false alarms and at the same time achieve early recognition of attacks, we will introduce a new detection function D(t) for the monitored coefficients, where, for simplicity, we denote, e.g. for the coefficient V(t), the function Y[V(t)>1,00] as YV(t):

$$D(t) = YV(t).YC(t) + YV(t).Y\rho(t) + YC(t).Y\rho(t)$$

The following figures show the course of the detection function for attacks S1 and N2.

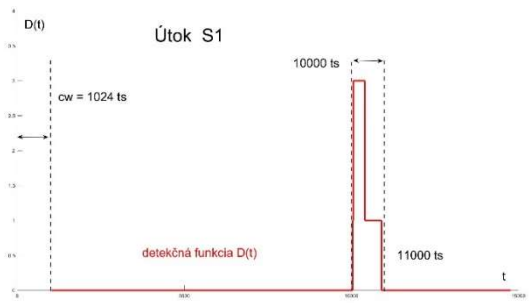


Fig. 51: S1, course of the detection functi

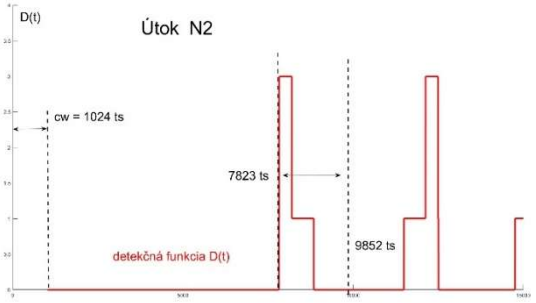


Fig. 52: N2, course of the detection function D(t)

Thanks to the shape of the function D(t), we have eliminated possible false alarms, and at the same time, based on the values 1 or 3, we can recognize at what time two or all 3 statistical parameters detected the attack. Table 3 presents the evaluation of the use of the detection function D(t) for the analysed attacks. The first value represents the number of false alarms, the second determines the time when at least two coefficients detected the attack, and the third value indicates the time when all three statistical coefficients V(t), c(t) and ρ(t) detected the attack.

attack	S1	N2	LR3	S5	N6	N7	N8
D(t)	0 – 34 - 52	0 - 5 - 5	x	x	0 – 75 - 119	0 - 228 - 480	0 - 47 - 210

Tab. 3: Evaluation of DDoS attack recognition using the detection function D(t)

Recognition of the attack based on the limit value was effective mainly for standard attacks, while in several cases two coefficients exceeded their limit values substantially earlier than all three. For non-standard attacks such as LR3, P4 and S5, no two detection intervals, if they arose at all, had a common intersection, and therefore the attack was not recognized. We will also apply the same methodology to attack recognition using the predictive tunnel. In the previous chapter, taking into account the attack recognition time and the number of false alarms, we determined the three most effective statistical coefficients: skewness K(t), kurtosis Skw(t) and variability V(t). Since we selected coefficients whose values increase significantly at the onset of a DDoS attack, we will deal only with the upper limit of the predictive tunnel. We will evaluate the time of exceeding the upper limit as a detection interval and describe it with a two-valued function Γ[.], where, e.g. for the coefficient of kurtosis, we denote ΓK(t) = Γ[K(t)>Hr(t)], where Hr(.) is the upper limit of the prediction interval for the coefficient K(.).

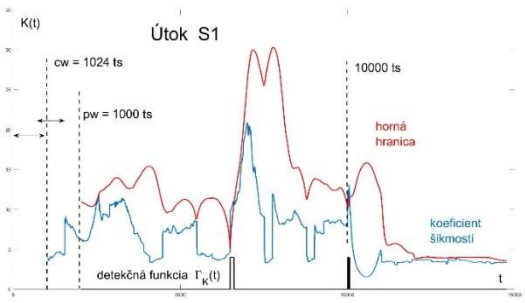


Fig. 53: S1, course of the function Γ(t)

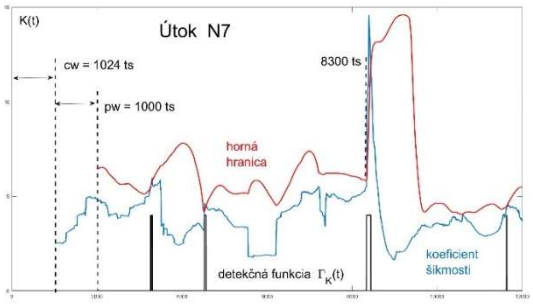


Fig. 54: N7, course of the function Γ(t)

Similarly to the previous method, recognition by means of a threshold value, we now also introduce a detection function that will be a combination of the identifiers Γ(t):

$$DP(t) = ΓK(t).ΓSkw(t) + ΓK(t).ΓV(t) + ΓSkw(t).ΓV(t)$$

Table tab.2 presents the evaluation of the use of the detection function DP(t).

attack	S1	N2	LR3	S5	N6	N7	N8
$D_P(t)$	1 – 6 - 28	0 - 1 - 1	1 - 1 - 1	1 - 18 - 41	1 - 12 - 12	1 - 39 - 49	0 - 1 - 1

Tab. 4: Evaluation of DDoS attack recognition using the detection function D(t)

Evaluation of the tests

On the basis of the experiments carried out so far, we can see that the use of selected statistical coefficients has considerably greater potential for fast recognition of DDoS attacks (FAD – Fast Attack Detection) than standard monitoring of average and peak intensity. In a DDoS attack, there is a strong assumption that these intensities will increase significantly. However, the average intensity grows only linearly depending on the size of the sliding window, which leads to late detection of the attack, and recognizing an attack by setting some threshold for the peak rate can lead to frequent false alerts owing to the occurrence of peaks in the IP flow.

When standard IP traffic turns into a DDoS attack, not only do the average and peak intensities change significantly, but the overall probabilistic structure of the flood flow also changes. To capture this change, we used the statistical coefficients presented. These can be divided into two types: coefficients that exceeded some detection value at the onset of the attack, and coefficients that responded to the attack with a marked change in their values or with an impulse. Some coefficients even belong to both groups. On the basis of this division, we created:

- The method of using detection limits (MDH)
- The method of using prediction tunnels (MPT)

By comparing Tables 3 and 4, we can see that MPT recognized every considered type of attack, and did so substantially earlier than MDH; however, for most attacks it signalled one false alarm. We assume that this problem could be eliminated in several ways, for example by adding another suitable detection coefficient, but by some type of smoothing of the coefficient values before using the predictive tunnel, e.g. exponential or Holt smoothing.

1.3 Detection of malicious code (Malware)

Malware, short for malicious software, comprises various types of malicious software, including viruses, worms, Trojan horses and ransomware. Each type has unique characteristics and methods of infection and damage. Viruses, for example, attach themselves to clean files and spread uncontrollably, damaging file systems. Worms, on the other hand, replicate themselves without human intervention and often exploit network vulnerabilities. Trojan horses deceive users by disguising themselves as legitimate software, thereby gaining unauthorized access to systems.

Ransomware represents a particularly dangerous form of malware. It encrypts the victim's data and demands a ransom for the decryption key. This type of malware can cause considerable operational disruption and financial losses. The impact of malware is not limited to individual users; it also extends to large organizations and can lead to massive data breaches, financial losses and reputational damage.

Preventing and mitigating malware attacks involves a combination of robust anti-virus software, user education and adherence to cybersecurity best practices, such as regular software updates and caution when downloading files or applications from the internet. Understanding the various forms of malicious software is crucial for effective cybersecurity measures, as it enables individuals and organizations to recognize potential threats and respond accordingly.

From the broader spectrum of malware, we decided to focus our tests only on the *"Trojan horse"* type. The tests were performed on the freely available *Trojan Detection* dataset for the following methods, which are among the most frequently used AI/ML algorithms and methods in this field:

- Random Forest
- AdaBoost
- Naive Bayes
- FNN-type neural network (Feed-Forward Neural Network)

The data in the *Trojan Detection* corpus come from the Canadian Institute for Cybersecurity (CIC) and contain records of flows in a computer network. A packet flow here denotes a series of packets in the communication of two applications that share the same group of data – source and destination IP address, source and destination port and protocol type. The dataset contains records of two types:

- malicious – trojan, 90683 samples of malicious flows generated by Trojan horses
- harmless – benign, 86799 samples of harmless benign network traffic

In the dataset, the following 86 parameters or features are recorded for each flow:

Flow ID	Source IP	Source Port
Destination IP	Destination Port	Protocol
Timestamp	Flow Duration	Total Fwd Packets
Total Backward Packets	Total Length of Fwd Packets	Total Length of Bwd Packets
Fwd Packet Length Max	Fwd Packet Length Min	Fwd Packet Length Mean
Fwd Packet Length Std	Bwd Packet Length Max	Bwd Packet Length Min
Bwd Packet Length Mean	Bwd Packet Length Std	Flow Bytes/s
Flow Packets/s	Flow IAT Mean	Flow IAT Std
Flow IAT Max	Flow IAT Min	Fwd IAT Total
Fwd IAT Mean	Fwd IAT Std	Fwd IAT Max
Fwd IAT Min	Bwd IAT Total	Bwd IAT Mean

Bwd IAT Std	Bwd IAT Max	Bwd IAT Min
Fwd PSH Flags	Bwd PSH Flags	Fwd URG Flags
Bwd URG Flags	Fwd Header Length	Bwd Header Length
Fwd Packets/s	Bwd Packets/s	Min Packet Length
Max Packet Length	Packet Length Mean	Packet Length Std
Packet Length Variance	FIN Flag Count	SYN Flag Count
RST Flag Count	PSH Flag Count	ACK Flag Count
URG Flag Count	CWE Flag Count	ECE Flag Count
Down/Up Ratio	Average Packet Size	Avg Fwd Segment Size
Avg Bwd Segment Size	Fwd Header Length.1	Fwd Avg Bytes/Bulk
Fwd Avg Packets/Bulk	Fwd Avg Bulk Rate	Bwd Avg Bytes/Bulk
Bwd Avg Packets/Bulk	Bwd Avg Bulk Rate	Subflow Fwd Packets
Subflow Fwd Bytes	Subflow Bwd Packets	Subflow Bwd Bytes
Init_Win_bytes_forward	Init_Win_bytes_backward	act_data_pkt_fwd
min_seg_size_forward	Active Mean	Active Std
Active Max	Active Min	Idle Mean
Idle Std	Idle Max	Idle Min
Class – Trojan/Benign		

Tab. 5: Group of features in the Trojan detection dataset

Preparation and testing procedure

Work with the models consisted of the following steps.

1. Data preprocessing
   - Loading the dataset for Trojan detection using the pandas library.
   - Data cleaning: omission of irrelevant and duplicate records, supplementing missing data as needed.
   - Encoding of categorical data (label encoding): conversion of categorical data such as the flow identifier (Flow ID), source and destination IP address (Source IP, Destination IP) and timestamp (Timestamp) – into numerical values.
   - Class Label Encoding: Conversion of the class labels ("Trojan" and "Benign") into binary form (1 and 0, respectively).
   - Feature Selection: Use mutual information to select the features that are most important for classification
2. Splitting the data into training and test data
   - Splitting the data into a training set and a test set (typically 80% training and 20% test data). The function using train_test_split from the scikit-learn library was used for the split.
3. Computations using machine learning (Machine Learning) algorithms: Random Forest, AdaBoost (adaptive boosting), Gaussian Naive Bayes and Deep learning with a feed-forward neural network. A description of the machine learning algorithms used follows.

The analysis and evaluation of the tests were carried out in the same way as for the previous types of cyber threats, based on defined metrics such as *Precision, Recall, F-1 Score, Accuracy* quantified from the *confusion matrix*. The computations themselves were carried out in the Google Colab test environment.

Test results

The test results are clearly presented in Table 6.

Method	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Processing time (sec.)

Random Forest classifier (50 estimators)	99.99	0.999967	0.999967	0.999967	9 sec
trying different numbers of trees: 10	99.99				additional, in total 60 sec.
20	99.9934				
30	99.9967				
40	100.000				
50	99.9934				
60	99.9967				
70	99.9967				
80	99.9967				
90	99.9934				
AdaBoostClassifier (100 estimators)	99.9639				20 sec.
cross-validation to obtain a more robust estimate	99.9626				60 sec.
GaussianNB Classifier					20
gnb.score	91.4984				
cross_val_score	90.1080				
using reciver operating characteristic score	92.1831				
weighted avg		92.1035	91.4985	91.4487	
Deep learning (Feed forward neural network) – 10 epoch	98.3947				55 sec. training alone
weighted avg		98.3953	98.3947	98.3947	

Tab. 6: Test results for malware detection

Evaluation of the tests

Based on the experiments carried out, it can be concluded that the most suitable methods for detecting "Trojan horse" malware from network traffic are the methods

- Random Forest: success rate 99.99%, evaluation time ~9 s
- AdaBoost: success rate 99.96%, evaluation time ~19 s
- FNN-type neural network: success rate 98.39%, evaluation time ~15 sec.

Of these, Random Forest performed the computations twice as fast. The next one in order, the FNN-type neural network, still has a high success rate (98.39%), but the network training time was longer (55 sec.). The fastest method was the GaussianNB Classifier – however, it achieved the worst results (91.49%). We consider the Random Forest algorithm to be the most suitable method for detecting "Trojan horse" malware by analysing traffic.

1.4 Phishing detection

Phishing is a deceptive cyber tactic used by attackers to obtain sensitive information from unsuspecting victims. This method usually involves sending fraudulent e-mails or messages that imitate legitimate sources, such as banks, government agencies or well-known companies. The aim is to deceive recipients and make them disclose personal data, such as login credentials, credit card numbers or other sensitive information.

These attacks can be highly sophisticated and use social engineering techniques to create a sense of urgency or fear, which forces victims to react quickly without questioning the authenticity of the request. Phishing messages often contain links to fake websites that closely resemble legitimate ones, where victims unknowingly enter their sensitive information.

Preparation and testing procedure

Testing was carried out on a dataset consisting of a file with 57,000 logs obtained from traffic at FRI UNIZA, originally containing approximately 160 variables, of which 41 were identified as relevant for identifying phishing. Several of these variables were categorical, and it was therefore necessary to preprocess them using a combination of one-hot encoding and the hashing trick, which expanded the dataset to 2,009 variables suitable for clustering.

Before clustering, all variables were standardized using StandardScaler in order to remove differences in value ranges. Subsequently, PCA dimensionality reduction to 50 components was applied, which allowed efficient work with the clustering algorithms while preserving the variability of the data.

This step was necessary and time-consuming, and it is of fundamental importance for the preparation of the machine learning model. Logs originating from different systems contain different variables, and therefore, when creating the final model for identifying phishing, it will be a major challenge to develop a robust script capable of processing unexpected or new variables as well.

From the wider spectrum of AI/ML methods and algorithms used for phishing detection, we decided to test the following clustering methods:

- K-means
- HDBSCAN

Test results

1.4.1.1 k-means

When applying the K-Means algorithm to our log dataset, we initially assumed 3 clusters. The resulting distribution was very uneven:

Cluster 0: 57,695 points

Cluster 1: 2 points

Cluster 2: 24 points

The K-Means results are visualized in Figures 55 and 56.

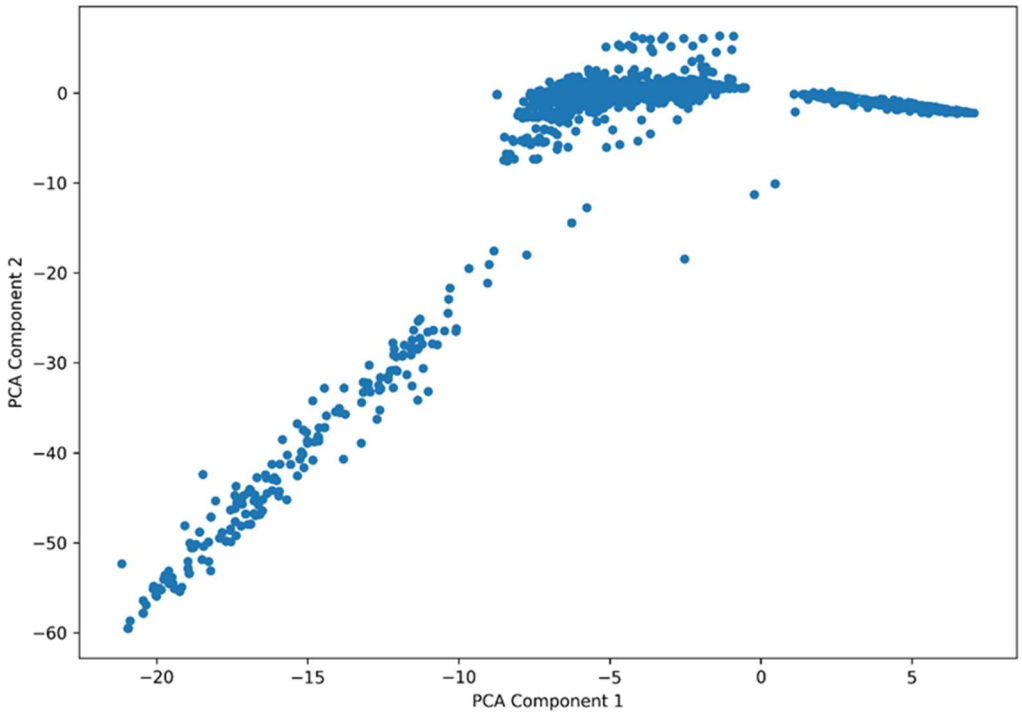


Fig. 55: 2-D visualization of k-means

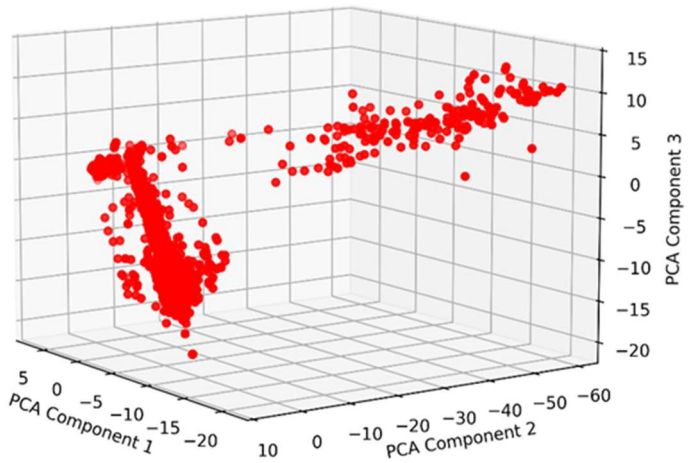


Fig. 56: 3-D visualization of k-means

Changing the assumed number of clusters (e.g. to 4 or 5) did not yield a better distribution of the data. Some new clusters contained only 1–2 points, while the main cluster still captured the majority of the data. This uneven distribution reflects the natural structure of the dataset, in which most logs are of a common type and only a very small number of logs represent suspicious or phishing events.

K-Means assigns every point to some cluster, without the possibility of labelling points as lying outside a cluster (outliers). As a result, extreme or non-standard values are automatically included in one of the clusters, often in small or marginal clusters. This may reduce the ability of K-Means to detect atypical or suspicious logs on its own.

Overall, it appears that K-Means is suitable for a fast and deterministic division of data into a predetermined number of clusters, but its ability to capture finer differences or identify deviations in logs is limited. The uneven distribution in this case corresponds to the natural imbalance of the data, in which the number of phishing logs is very low.

1.4.1.2 HDBSCAN

The HDBSCAN algorithm was applied to the log dataset with the parameters min_cluster_size=3000 and min_samples=20. These settings made it possible to identify 6 main clusters and at the same time to mark outliers that did not fall into any cluster.

The resulting distribution of the number of logs in the individual clusters was as follows:

Outliers (label = -1): 14,913 points

Cluster 0: 13,493 points

Cluster 1: 6,702 points

Cluster 2: 7,704 points

Cluster 3: 7,891 points

Cluster 4: 3,363 points

Cluster 5: 3,655 points

Interpretation of the results

1) Outliers (-1):
   - the algorithm marked 14,913 logs as outliers that did not belong to any dense cluster
   - these logs probably contain atypical or potentially suspicious events, which is an advantage for detecting phishing or anomalies
2) Main clusters (0–5):
   - the clusters are relatively evenly represented, with the largest clusters containing 13,493 and 7,891 points and the smaller clusters 3,363 and 3,655 points
   - this more even distribution compared with K-Means makes it possible to better capture finer differences between logs and to identify the density structure in the data
3) Meaning of the parameters:
   - min_cluster_size=3000 ensured that only significant clusters containing a sufficient number of points are formed
   - min_samples=20 determined the sensitivity to density and contributed to the detection of outliers

The HDBSCAN results are visualized in Figures 57 and 58.

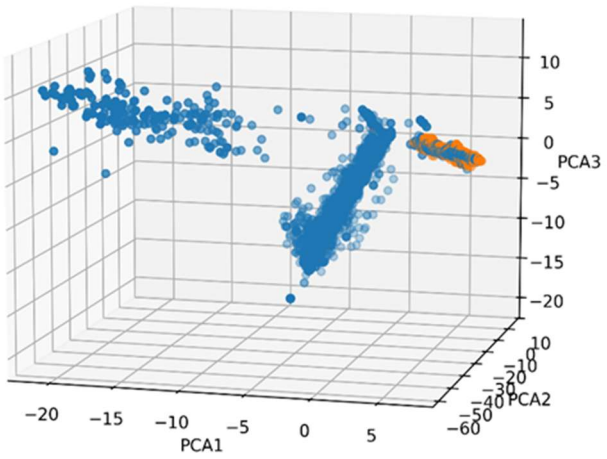


Fig. 57: 3-D visualization of HDBSCAN

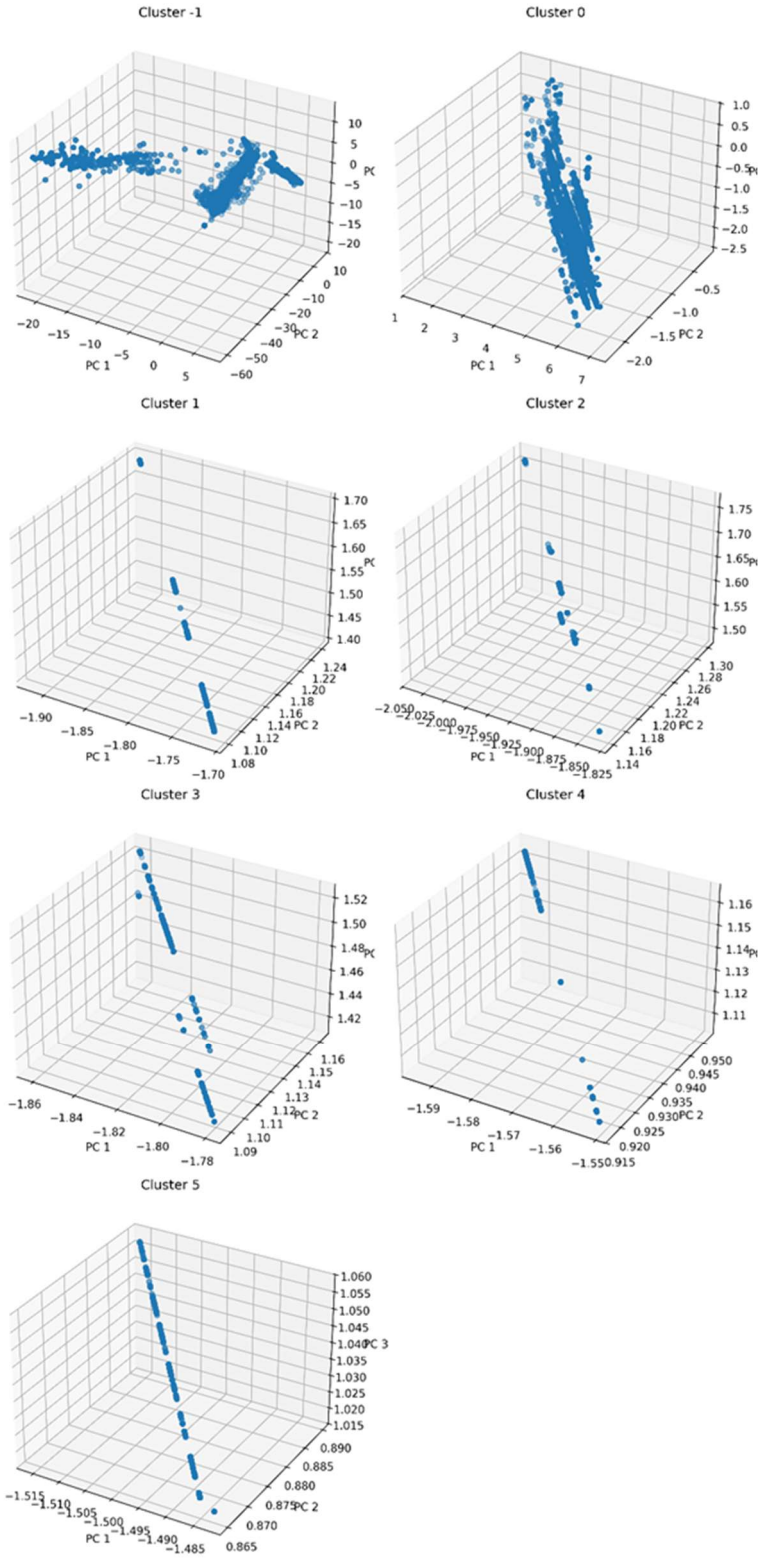


Fig. 58: 3-D visualization of the individual clusters in HDBSCAN

In Figure 58, we can see that the logs assigned to the individual clusters are distributed in parallel planes, while the logs marked as outlying values (outliers) are located outside these main planes. This visual effect indicates that HDBSCAN can effectively distinguish typical behaviour from atypical or suspicious events, unlike K-Means, which assigns every point to some cluster.

In this context, using the distance of a log from the plane of the main cluster as a probabilistic phishing score represents a new and innovative way of interpreting the data. This approach makes it possible to quantify the degree of deviation of individual logs and potentially to identify suspicious or malicious events with finer granularity than the usual binary labelling of outliers.

At the same time, this concept needs to be verified in further research in order to confirm its effectiveness in the automated detection of anomalies and phishing attacks, and to make it possible to develop a robust system that combines the distance score with other log characteristics (e.g. time, IP addresses, event type) for more accurate identification of suspicious activities.

Evaluation of the tests

HDBSCAN has proved to be a suitable tool for analysing logs with an uneven distribution, as it is able to:

- identify dense regions of logs
- mark outliers that may represent phishing or other suspicious events
- maintain a more even distribution of clusters compared with K-Means, which facilitates the interpretation of the results and further analysis
- a new and innovative approach: if a new log is assigned to one of the clusters, it is usually neither an anomaly nor a potential attack. However, if a log is not assigned to any cluster, its degree of severity can be quantified using the distance from the nearest cluster, thereby producing a probabilistic phishing score. This approach allows the risk of suspicious events to be quantified more finely and supports automated anomaly detection.

Overall, HDBSCAN provides a robust and flexible approach to clustering network logs, especially for large and heterogeneous datasets, while the new concept of probabilistic outlier scoring represents a promising area for further research and verification of its effectiveness.

K-Means is suitable if a fast and deterministic decomposition is needed; however, it does not detect outliers and distributes points unevenly in data of uneven density.

HDBSCAN is suitable for robust log analysis, as it can detect both dense regions and outlying values, which is a significant advantage in identifying phishing and other anomalies in network logs. Unlike algorithms that assign every point to a cluster, HDBSCAN makes it possible to mark atypical or outlying logs (outliers) that do not meet the density criteria required for a cluster to form. In this way, the algorithm effectively filters out noise and focuses on logs that represent repeatable and significant patterns of behaviour.

In the pilot testing, HDBSCAN proved more suitable for capturing finer differences between logs that could be indicators of suspicious or non-standard events. The algorithm is able to divide the data into clusters according to the density of points, while preserving the hierarchical structure and making it possible to identify smaller but significant groups of logs that could be “lost” in massive clusters when using classic K-Means.

The main advantage of HDBSCAN over K-Means is therefore its ability to:

- automatically determine the number of clusters according to data density

- robustly identify outliers that may indicate anomalies or phishing activities
- preserve fine differences between logs, which improves the interpretation and analytical value of the results.

Overall, HDBSCAN provides a flexible and robust approach to log analysis that takes into account the natural variability of the data and enables effective detection of suspicious events without the need for a predetermined number of clusters.

1.5 Detection of advanced persistent threats APT

Advanced persistent threats (APT) represent a category of cyber attacks in which an unauthorized user gains access to a network and remains undetected for an extended period of time. These attacks are usually aimed at stealing data or monitoring network activity rather than causing immediate damage. APTs are often sponsored by nation states or large criminal organizations, which makes them highly sophisticated and difficult to detect and remove. APTs involve a lengthy, stealthy and complex infiltration process. Attackers use various methods to gain entry, including spear-phishing, exploitation of vulnerabilities or the use of malicious software. Once inside, they establish a foothold, often create backdoors and use techniques to remain undetected. The ultimate goal of an APT is not immediate disruption, but long-term espionage, data theft or the gaining of strategic advantages.Combating APTs requires a multifaceted approach, including the implementation of multi-layered security defences, continuous monitoring of network activity, regular security audits and training employees to recognize potential threats. Early detection and rapid response are essential to mitigate the impact of APTs. Organizations must adopt a stance of constant vigilance and readiness to respond to these advanced threats.

Testing was carried out using the freely available *DAPT2020* dataset (Dataset for Advanced Persistent Threats (APT) 2020). The DAPT2020 dataset (Dataset for Advanced Persistent Threats (APT) 2020) is a cybersecurity dataset specifically designed for research in the field 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 goal 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 kill chain, which represents sophisticated APT campaigns, and provides data for the development of detection systems for lateral movement, privilege escalation, data exfiltration and command and control C2 (command & control) activities.

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

- Network topology: A small enterprise network with multiple operating systems (Windows 10, Windows Server, Ubuntu) and standard services (Active Directory, web servers, file shares) was simulated.
- Attack simulation: The research team ran real malware and APT attack tools according to the documented tactics, techniques and procedures TTP (Tactics, Techniques, and Procedures) of known APT groups.
- Benign background traffic: To create a realistic mix of data, ordinary 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 - nmap, sqlmap, vulnerability scanning, ...
- Day 3: Establishing a foothold - reverse shells, SQL exploitation, authentication bypass
- Day 4: Lateral movement - internal network scanning, exploiting vulnerabilities
- Day 5: Data exfiltration - FTP transfers, uploads to Google Drive, C&C communication

The corpus contains 79 network flow parameters/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): Complete network traffic from the test environment, enabling 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 key to detecting lateral movement, privilege escalation and persistence
  - Sysmon logs - detailed process creation, network connection and file modification events (invaluable for forensic tracking)
- host-based artefacts
  - memory dumps - for the purposes of malware analysis
  - disk images - for the purposes of forensic investigation
  - process listings and registry overviews
- alert and detection logs - outputs from security tools such as Suricata (IDS) and Elastic Security (SIEM) deployed during the experiment

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

1. It focuses on data theft and leakage. It includes an initial compromise attack, lateral movement on a database server, data staging and leakage via covert channels.
2. It focuses on ransomware deployment. It includes establishing a foothold, escalating privileges, disabling defences and deploying ransomware in the network.

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

The dataset has the following key properties:

- uses modern TTPs and malware samples from the 2019 – 2020 time frame
- captures attacks that take place over 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
- synchronous collection of network and host records enables correlation across sources, which is a necessity for APT hunting.

From the wider group of AI/ML algorithms and methods that can be used to detect persistent APT threats, we decided to test the following:

- Random Forest
- Support Vector Machine SVM
- Logistic regression

These methods are among the most frequently used techniques with a very good detection success rate.

Evaluation of tests

The evaluation of the tests was carried out in the same way as for the detection of unsolicited SPAM messages described in chapter 1.1 Detection of unwanted e-mail messages (Spam), using the same evaluation metrics and criteria.

Test results

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. 7: Test results for APT threat detection

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

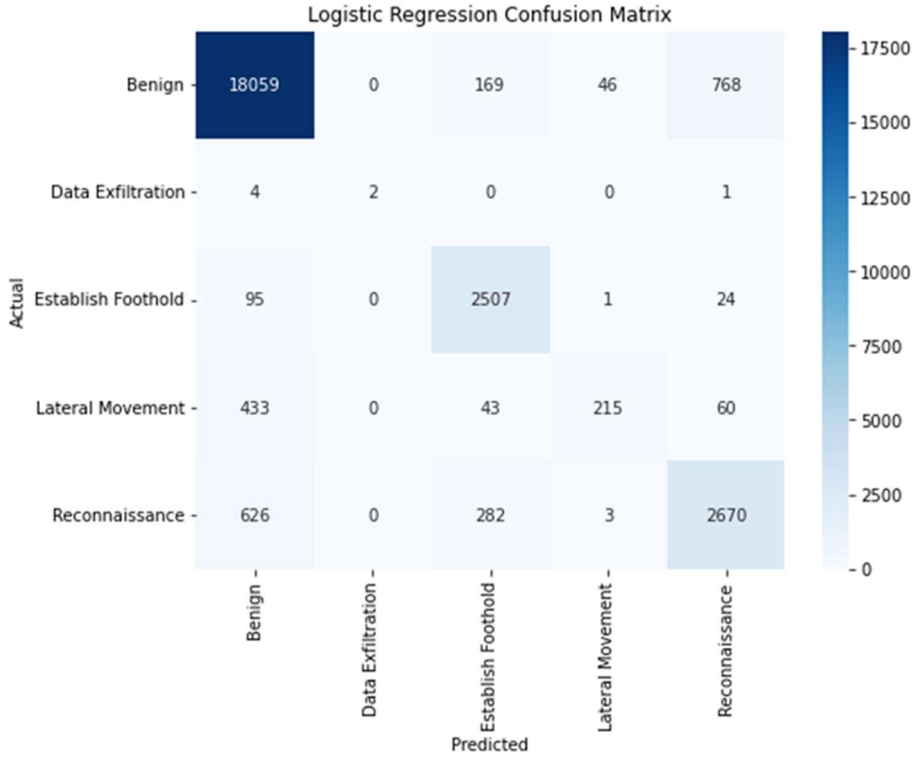


Fig. 59: Confusion matrix for logistic regression



Fig. 60" Confusion matrix for SVM

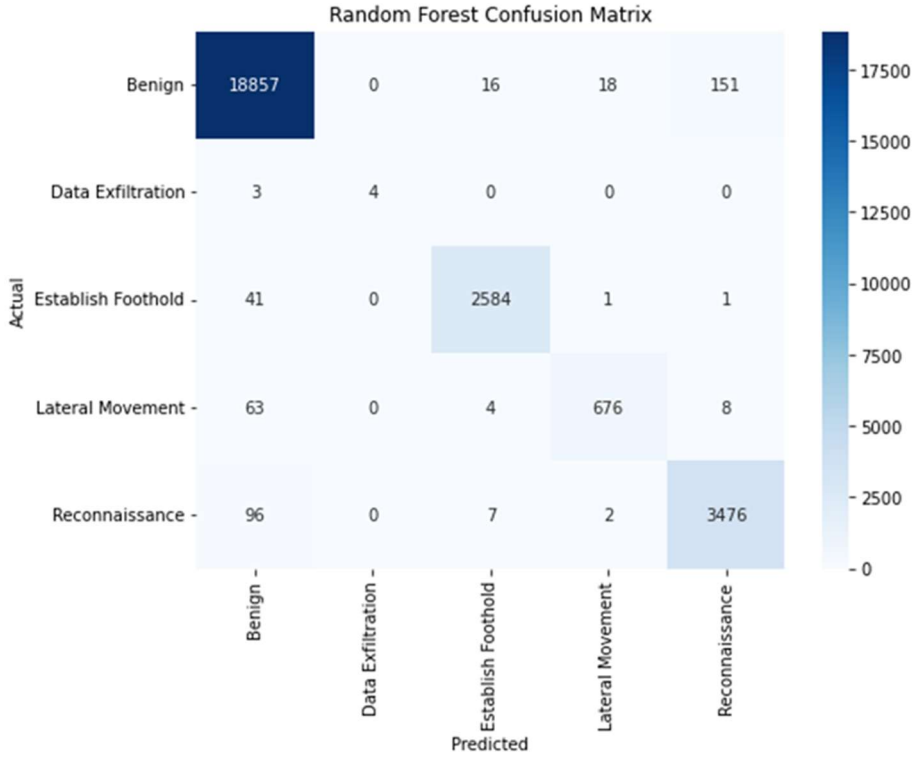


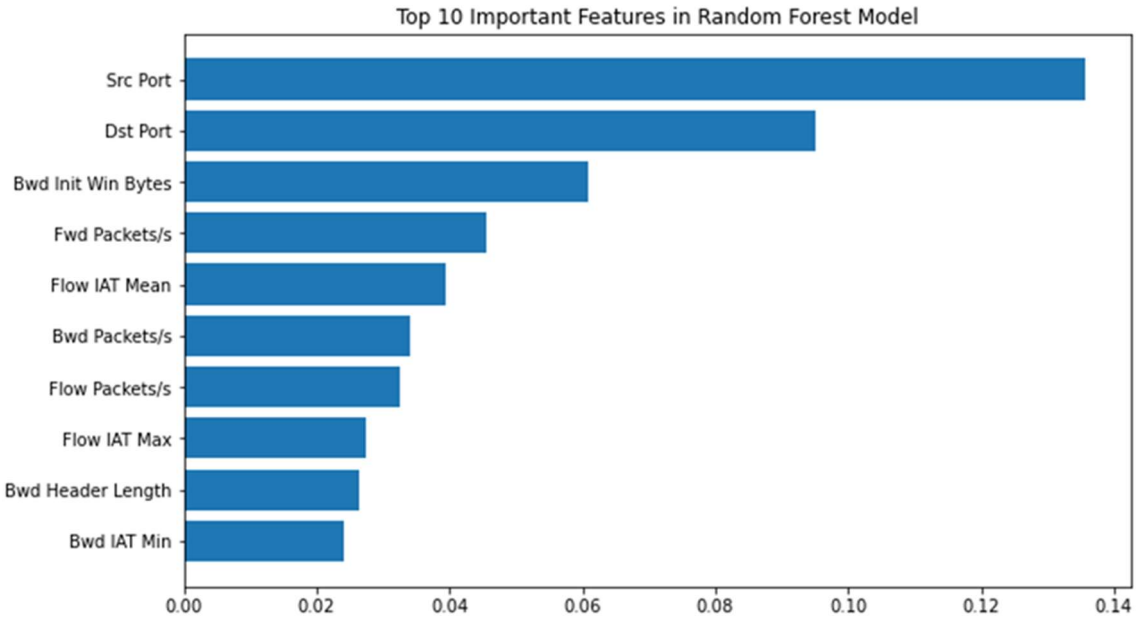
Fig. 61: Confusion matrix for Random Forest

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



Fig. 62: Heat map of the correlation coefficient for selected features

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



Funded by the European Union
NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

Fig. 63: Top 10 important features in the Random Forest model

The following figure visualizes the success rate (the Accuracy parameter) of the individual selected classification methods.

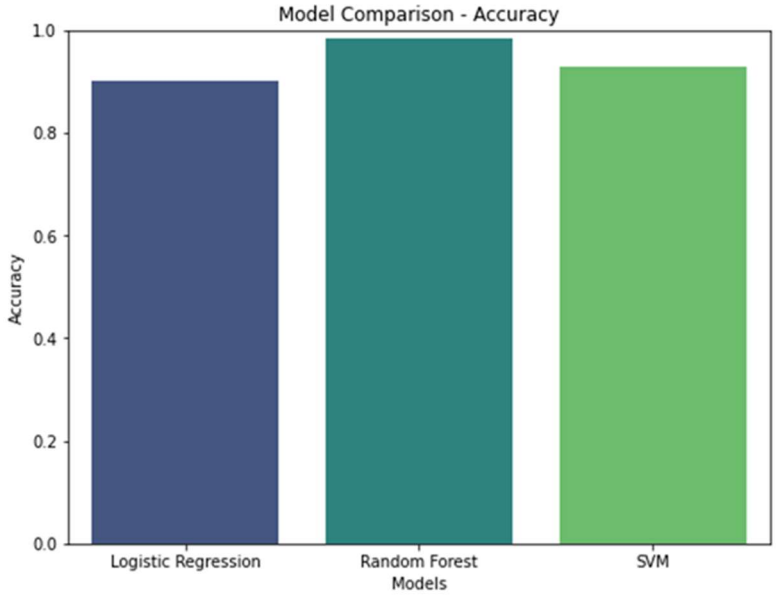


Fig. 64: Comparison of the success rate of the individual models

Evaluation of the tests

Based on the experiments performed, it can be concluded that the most suitable method for detecting advanced APT attacks is the most suitable method Random forest. 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 with a value of ~90%. In terms of the time needed for testing data, the Random forest and logistic regression methods achieve very comparable results at the level of tens of milliseconds, so both methods can be deployed for real-time applications. The SVM method needs a significantly longer detection time during testing, at the level of tens of seconds, which represents a time more than 100 times longer.

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

2 COMPONENT TESTS IN THE INNOSOC POC

BRAIN:IT

FRI

2.1 SIEM - ELK Stack

2.2 Shuffle

2.3 Wazuh

2.4 Greenbone Community edition

2.5 OpenCTI

2.6 MISP

2.7 GLPI

2.8 Netbox

Tested component: NetBox

2.8.1 TS-NB-01 Verification of the availability and basic functionality of NetBox

Test identifier: TS-NB-01

Test name: Verification of the availability and basic functionality of NetBox

Test type: Functional test

Test purpose: To verify whether NetBox is correctly deployed in the PoC environment, accessible via the web interface and ready for basic user and administrative work.

Subject of verification: The subject of the test is the verification of the basic operational readiness of the NetBox component after its deployment in the OpenStack environment. The test focused on whether the service is available, whether it is possible to access the web interface, authenticate successfully and work with the basic parts of the system without any obvious error or malfunction. It was also verified whether NetBox is ready for further use within the PoC as a recording and context tool.

Initial state: NetBox is deployed as a standalone service in the OpenStack environment. The system had a basic configuration and its application components were made available, including the database and supporting layer needed for it to run. Before the start of the test, it was assumed that the service was running, had a network connection assigned and was accessible via the defined web interface.

Test procedure:

1. The network availability of NetBox in the deployed environment was verified.
2. The web interface of the system was opened via a web browser.
3. The correct loading of the login or home screen of the system was verified.
4. A login was performed using an available user account.
5. After logging in, the loading of the basic parts of the interface and navigation in the system were verified.
6. The availability of selected recording parts of the system was checked, in particular devices, sites, addressing and other basic objects.
7. It was verified whether the system shows any obvious errors during ordinary work in the interface.

Expected result:

The expected result is that NetBox will be accessible in the PoC environment via the web interface, will allow a successful login and will provide a functional user interface for basic work with the system. It was also expected that the main parts of the application needed for recording and managing infrastructure objects would be available.

Actual result

NetBox was available in the test environment and its web interface loaded correctly. Logging in to the system was successful and, after authentication, it was possible to access the basic parts of the application. The availability of selected recording sections of the system and basic navigation in the user interface were verified. No critical errors that would prevent further use of the system within the PoC were observed during the test.

Test evaluation: Passed.

Based on the verification performed, it can be concluded that NetBox was successfully deployed and put into operation to the extent necessary for the PoC solution. The test confirmed that the component is available, functional and ready for basic recording and context use within the proposed solution.

Findings and notes

Test TS-NB-01 verified only the basic availability and functionality of the system, not all of its extended capabilities or integration links to the other PoC components. The result therefore confirms the operational readiness of NetBox at the level of a standalone service, not the completeness of its involvement in all planned scenarios of the solution. The test confirmed the prerequisite for further tests focused on object recording, infrastructure context and the practical use of NetBox within the PoC.

Conclusion

The test demonstrated that NetBox is available and basically functional in the PoC environment. This confirmed its readiness for further verification in the area of recording infrastructure objects and use as a supporting source of context data for the other parts of the solution.

2.8.2 TS-NB-02 Verification of the recording of infrastructure objects

Test identifier: TS-NB-02

Test name: Verification of the recording of infrastructure objects

Test type: Functional test

Test purpose: To verify whether NetBox enables the recording and management of basic infrastructure objects relevant to the PoC solution, in particular sites, devices, address spaces and selected network data.

Subject of verification: The subject of the test is the verification of whether NetBox, after deployment and basic configuration, is usable for the practical recording of infrastructure objects. The test focused on whether it is possible to create, display and edit selected objects in the system and whether these data are correctly stored and available in the system. It was also verified whether the created data model meets the needs of the experimental PoC solution.

Initial state: NetBox is deployed and functional in the OpenStack environment. The system is accessible via the web interface and the user has permission to work with the recorded objects. The basic categories and structures needed for recording infrastructure data are prepared in the system..

Test procedure:

1. The availability of the system sections intended for recording infrastructure objects was verified.
2. A site was created in the system or its existence was verified.
3. A device assigned to a site or logical location was created or verified.
4. The recording of an address space, prefix or IP address was verified.
5. The possibility of saving and subsequently displaying the created data was checked.
6. The possibility of editing a selected object was verified.
7. It was assessed whether the system provides usable object recording for the needs of the PoC..

Expected result: The expected result is that NetBox will enable the recording of basic infrastructure objects, their correct storage, display and basic management. It is also expected that the recorded data will be usable as basic infrastructure context within the PoC solution.

Actual result: It was possible to work with basic infrastructure objects in the system, in particular with sites, devices and address spaces. Their creation, display and basic editing were verified. The data were correctly stored in the system and available via the user interface. Within the scope of the test performed, it was confirmed that NetBox is usable as a tool for the basic recording of infrastructure data relevant to the PoC.

Test evaluation: Met.

Based on the verification performed, it can be concluded that NetBox in the PoC environment meets the requirement for the basic recording of infrastructure objects. The test confirmed that the system is usable for managing data on selected assets and addressing to the extent necessary for the experimental solution..

Findings and notes

Test TS-NB-02 focused on the basic recording functionality of the system, not on complete modelling of the entire infrastructure or on the integration of all object relationships. The verification therefore confirms the functionality of recording within the scope of the PoC, not the completeness or final operational maturity of the data model. The test also created the prerequisite for further verification of relationships between objects and the use of NetBox as a source of infrastructure context..

Conclusion

The test showed that NetBox is functional in the PoC environment as a recording tool for basic infrastructure objects. This confirmed its usability for further verification of logical relationships and contextual use in the proposed solution..

2.8.3 TS-NB-03 Verification of relationships and navigation in the infrastructure context

Test identifier: TS-NB-03

Test name:

Test type:

Test purpose:

Subject of verification: .

Initial state: .

Test procedure:

1. .

Expected result:

.

Actual result

.

Test evaluation

Test result: Passed.

.

Findings and notes

.

Conclusion

.

2.8.4 TS-NB-04 Verification of the use of NetBox as a source of context for the PoC

Test identifier: TS-NB-02

Test name:

Test type:

Test purpose:

Subject of verification: .

Initial state: .

Test procedure:

1. .

Expected result:

.

Actual result

.

Test evaluation

Test result: Passed.

.

Findings and notes

.

Conclusion

2.8.5 Evaluation of the tests

Test	Area verified	Result	Status
TS-NB-01	Service deployment	the service was successfully deployed	passed
TS-NB-02	Web interface	available and functional	passed
TS-NB-03	Object recording	basic objects created	passed
TS-NB-04	Context for the PoC	usable mainly manually	partially passed
TS-NB-05	Automated integration	was not fully completed	partially passed

2.9 DFIR IRIS

2.10 Kibana (ELK)