Harmonizing People, Processes, and Technology for Robust Cybersecurity

D4.3 Integrated Cybersecurity Monitoring Suite v2

Document Summary Information

Grant Agreement No	101158555	**Acronym**	CYberSynchrony
Full Title	Harmonizing People, Processes, and Technology for Robust Cybersecurity		
Start Date	01/06/2024	**Duration**	36 months
Project URL	https://www.cybersynchrony.eu/		
Deliverable	D4.3 Integrated Cybersecurity Monitoring Suite v2		
Work Package	WP4 – Advanced Proactive Cyber Risk Monitoring and Rapid Response (CYBRITE)		
Contractual due date	31/05/2026	**Actual submission date**	30/06/2026
Type	Other	**Dissemination Level**	Public
Lead Beneficiary	Maggioli (MAG)		
Responsible Author	Vasilis Keleris (MAG)		
Contributions from	Vasilis Keleris (MAG), Evangelos Michos (IANUS), Leandros Maglaras (TRUST)		

Funded by the European Union

ECCC

Funded by the European Union under Grant Agreement no. 101158555 and supported by the European Cybersecurity Competence Centre (ECCC). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Cybersecurity Competence Centre. Neither the European Union nor the granting authority can be held responsible for them.

Revision history

Version	Issue Date	Changes	Contributor(s)
0.1	25/02/2026	Initial deliverable structure	Vasilis Keleris (MAG)
0.2	20/03/2026	Integration of deliverable ToC and scoping. Chapters 1, 2	Vasilis Keleris (MAG)
0.3	30/04/2026	Technical contributions - Wazuh, Suricata, Anomaly Detector IDS	Vasilis Keleris (MAG), Evangelos Michos (IANUS)
0.4	19/05/2026	Technical contributions - WEKA, trustSENSE	Leandros Maglaras (TRUST)
0.5	29/05/2026	Final draft for internal review	Vasilis Keleris (MAG), Evangelos Michos (IANUS), Leandros Maglaras (TRUST)
0.6	10/06/2026	Peer review	George Karavokiros (BYTE), Evangelos Michos (IANUS)
0.7	19/06/2026	Final draft addressing comments from internal review	Vasilis Keleris (MAG), Evangelos Michos (IANUS), Leandros Maglaras (TRUST)
0.8	23/06/2026	QA review	Eleni Keliri (eBOS)
1.0	29/06/2026	Final version ready for submission	Vasilis Keleris (MAG)

Disclaimer

The content of the document herein is the sole responsibility of the publishers and it does not necessarily represent the views expressed by the European Commission or its services.

While the information contained in the documents is believed to be accurate, the authors(s) or any other participant in the CYberSynchrony consortium make no warranty of any kind with regard to this material including, but not limited to the implied warranties of merchantability and fitness for a particular purpose.

Neither the CYberSynchrony consortium nor any of its members, their officers, employees or agents shall be responsible or liable in negligence or otherwise howsoever in respect of any inaccuracy or omission herein.

Without derogating from the generality of the foregoing neither the CYberSynchrony consortium nor any of its members, their officers, employees or agents shall be liable for any direct or indirect or consequential loss or damage caused by or arising from any information advice or inaccuracy or omission herein.

Copyright message

© CYberSynchrony Consortium This Deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both. Reproduction is authorised provided the source is acknowledged.

- 3 months before Deliverable's Due Date: Table of Contents ready to be reviewed by WP leader/ Technical Manager/ Quality Manager
- 2 months before Deliverable's Due Date: 50% should be complete. Review by the Quality Manager (& Technical Manager for software).
- 1 month before Deliverable's Due Date:100% should be complete. Review by the Quality Manager and Peer Reviewers.
- M: Sent to PC for Submission to the EC after addressing all comments by Quality Manager and Peer Reviewers

© CYberSynchorny, 2026

Table of Contents

Executive Summary 8
1 Introduction 9
1.1 Mapping CYberSynchrony Outputs 9
1.2 Linkage to other project outputs 9
1.3 Deliverable Structure 9
2 Position in CYberSynchrony Architecture 11
3 Suricata IDS (IANUS) 12
3.1 Updates since V1 12
3.2 Integration, interoperability and deployment 12
3.2.1 Deployment Configuration 13
3.2.2 Indicative example 14
3.2.3 Integration with Wazuh 15
3.2.4 Communication with other platform components 16
4 Wazuh (IANUS) 19
4.1 Updates since V1 19
4.2 Integration, interoperability and deployment 19
4.2.1 Accessing the Manager pod 20
4.2.2 Suricata IDS Alert example 21
4.2.3 Wazuh SCA Compliance Alert 22
4.2.4 Wazuh Rootcheck Alert 23
4.2.5 Communication with other platform components 24
4.2.6 Wazuh Dashboard 25
5 Anomaly detector IDS 30
5.1 Updates since V1 30
5.2 Integration, interoperability and deployment 30
5.2.1 Role within the CYBRITE monitoring chain 30
5.2.2 Event ingestion via Kafka 31
5.2.3 Runtime deployment 32
5.2.4 Monitoring dashboard 32
6 WEKA 36
6.1 Updates since V1 36
6.2 Integration, interoperability and deployment 36
7 trustSENSE 39
7.1 Updates since V1 39
7.2 Integration, interoperability and deployment 39
8 KPIs Mapping 42
8.1 KPI1.1: Number of Threat Assessments Carried Out 42
9 Conclusions 44

List of Figures

Figure 1. Positioning of D4.3's components in the CYberSynchrony platform 11
Figure 2. Wazuh-Suricata-MISP integration and data flow 12
Figure 3. Vulnerability Dashboard in Wazuh 25
Figure 4. Vulnerabilities discovered in Dashboard 26
Figure 5. Wazuh Threat Hunting Dashboard 26

© CYberSynchorny, 2026

Figure 6. Wazuh Threat Hunting Dashboard (Suricata events) 27
Figure 7. Wazuh Threat Hunting Dashboard (MISP events) 27
Figure 8. Wazuh Threat Hunting Dashboard (Wazuh events) 28
Figure 9. Wazuh Malware Detection Dashboard 28
Figure 10. Wazuh Malware Detection Events 29
Figure 11. Wazuh Malware Detection Event details 29
Figure 12. Kafka broker role in Inference pipeline 31
Figure 13. Detector's main dashboard 33
Figure 14. Network activity 34
Figure 15. Anomaly rate 34
Figure 16. Top talker IPs 35
Figure 17. Flows per protocol 35
Figure 18. WEKA GUI 36
Figure 19. Explorer interface 36
Figure 20. Sample data about weather 37
Figure 21. The Classify tab 38
Figure 22. The Visualize tab 38
Figure 23. Initial screen of trustSENSE 39
Figure 24. Assessment page with the 23 questions 39
Figure 25. Team's Trustworthiness Level Result 40
Figure 26. Example Mitigation actions proposed by the trustSENSE tool 41

List of Tables

Table 1: Adherence to CYberSynchrony's GA 9
Table 2: Suricata Deployment issues 13
Table 3: Suricata Fields 14
Table 4: Before and After Suricata integration 16
Table 5: Suricata communication with other components 16
Table 6: Wazuh Components 19
Table 7: Main fields of the Wazuh alert structure 20
Table 8: Main fields of the Wazuh rule object 21
Table 9: Wazuh communication with other tools 24
Table 10: Before and after integration 24
Table 11. Functional roles within the CYBRITE monitoring chain 31

© CYberSynchorny, 2026

Acronyms and abbreviations

Abbreviation / Term	Description
AD	Anomaly Detector
AE	Autoencoder
AI	Artificial Intelligence
API	Application Programming Interface
ARFF	Attribute-Relation File Format
CIS	Center for Internet Security
CSV	Comma-Separated Values
CVE	Common Vulnerabilities and Exposures
DCO	Defensive Cyber Operations
DNS	Domain Name System
EC	European Commission
ECCC	European Cybersecurity Competence Centre
ENISA	European Union Agency for Cybersecurity
ET	Emerging Threats
EVE	Event and Alert Output Format in Suricata
FIM	File Integrity Monitoring
GA	Grant Agreement
GDPR	General Data Protection Regulation
GUI	Graphical User Interface
HIPAA	Health Insurance Portability and Accountability Act
HMAC	Hash-based Message Authentication Code
HRM	Human Risk Management
HTTP	Hypertext Transfer Protocol
IDS	Intrusion Detection System
IoC	Indicator of Compromise
IP	Internet Protocol
ISO	International Organization for Standardization
JSON	JavaScript Object Notation

© CYberSynchorny, 2026

KPI	Key Performance Indicator
MHSA	Multi-Head Self-Attention
MISP	Malware Information Sharing Platform
MITRE ATT&CK	Adversarial Tactics, Techniques, and Common Knowledge
NIC	Network Interface Card
NIST	National Institute of Standards and Technology
PCI DSS	Payment Card Industry Data Security Standard
REST	Representational State Transfer
SCA	Security Configuration Assessment
SecOps	Security Operations
SIEM	Security Information and Event Management
SOC	Security Operations Centre
SOC2	System and Organization Controls 2
SSH	Secure Shell
SSO	Single Sign-On
Sysmon	Microsoft System Monitor/Sysinternals Sysmon
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UEBA	User and Entity Behaviour Analytics
UDP	User Datagram Protocol
URL	Uniform Resource Locator
WEKA	Waikato Environment for Knowledge Analysis
WP	Work Package
YAML	YAML Ain't Markup Language

© CYberSynchorny, 2026

Executive Summary

This deliverable is the second version of D4.1 "Integrated Cybersecurity Monitoring Suite v1". It reports the updated architecture, implementation status, and integration interfaces of the components developed in WP4 under tasks T4.1 (SIEM & Traffic Analysis Deployment) and T4.2 (Entity Behaviour Analytics for Behavioural Insight). Each component chapter is structured into two sections: updates since the first version, and integration, interoperability and deployment.

The monitoring suite spans two functional layers. The detection layer consists of Wazuh, operating as the SIEM for log collection and correlation, and Suricata IDS, responsible for network traffic inspection and structured event generation, both supported by Sysmon for host-level telemetry. These are complemented by the AI-based Anomaly Detector IDS, which processes Suricata flow events through a transformer-based autoencoder to identify behavioural deviations in network traffic. The analytics layer consists of WEKA, which applies machine learning techniques to historical incident data for post-forensic analysis, and trustSENSE, which supports human-in-the-loop risk assessment through user profile modelling and adversary behaviour analysis.

Pilot execution results and KPI assessment will be reported in WP8 deliverables.

© CYberSynchorny, 2026

1 Introduction

1.1 Mapping CYberSynchrony Outputs

This section presents the CYberSynchrony's Grant Agreement (GA) commitments, as extracted from the formal deliverable and task description, with respect to their outputs and work to be performed.

Table 1: Adherence to CYberSynchrony's GA

Component no	Component title	Respective Document Chapter(s)	Justification
DELIVERABLE			
D4.3	Integrated Cybersecurity Monitoring Suite v2	Chapters 1-7	This deliverable is the final update on the architecture, implementation and integration status of components developed under T4.1 and T4.2.
TASKS			
T4.1	Unveiling SIEM & Traffic Analysis Deployment	Chapters 3-5	T4.1 focuses on the deployment and design of SIEM systems and network traffic analysis tools. The described components—such as Wazuh, Suricata, and anomaly-based IDS—are aligned with this task's objective of enabling real-time monitoring and early threat detection
T4.2	Entity Behaviour Analytics (UEBA) Systems for Behaviour Insight	Chapters 6-7	T4.2 is represented through the WEKA module for post-forensic analysis, and the trustSENSE component. These tools support entity behaviour modelling, risk detection, and human-in-the-loop intelligence workflows.

1.2 Linkage to other project outputs

The present deliverable builds upon and aligns with D4.1. Linkage to other project outcomes has been reported in D4.1.

1.3 Deliverable Structure

This deliverable is structured into nine main chapters. Each chapter serves a specific role in presenting the technical development and integration of CYberSynchrony's monitoring components:

Chapter 1: Introduction

Outlines deliverable's objectives and how they align with the CYberSynchrony Grant Agreement. Includes:

© CYberSynchorny, 2026

- Mapping CYberSynchrony Outputs – alignment with GA components and project tasks.
- Linkage to other project outputs – dependencies on previous work.
- Deliverable Structure – document organization.

Chapter 2: Position in CYberSynchrony Architecture

Presents the updates on the positioning of the monitoring suite components within the CYberSynchrony platform architecture.

Chapter 3: Suricata IDS

Architecture, implementation status, and technology stack of the Suricata intrusion detection system, including integration interfaces.

Chapter 0:

Wazuh

Architecture, implementation status, and technology stack of the Wazuh SIEM, including its integration interfaces with other CYberSynchrony components.

Chapter 5: Anomaly detector IDS

Architecture, implementation status, and technology stack of the AI-based anomaly detection module, including integration interfaces.

Chapter 6: WEKA

Architecture, implementation status, and technology stack of the WEKA post-forensic analysis module, including integration interfaces.

Chapter 7: trustSENSE

Architecture, implementation status, and technology stack of the trustSENSE module, including integration interfaces.

Chapter 8: KPIs Mapping

Maps the monitoring suite components documented in Chapters 3–7 to the project KPIs they support, and outlines the measurement methodology and evidence sources.

Chapter 9: Conclusions

Summarizes the M25 status of the monitoring suite components, identifies open items, and points to the WP8 pilot validation activities where full operational evaluation will be reported.

© CYberSynchorny, 2026

2 Position in CYberSynchrony Architecture

This section describes how the components documented in this deliverable are positioned within the CYberSynchrony platform architecture. The architectural view presented here (Figure 1) updates the one in D4.1, reflecting the implementation and integration status at M25.

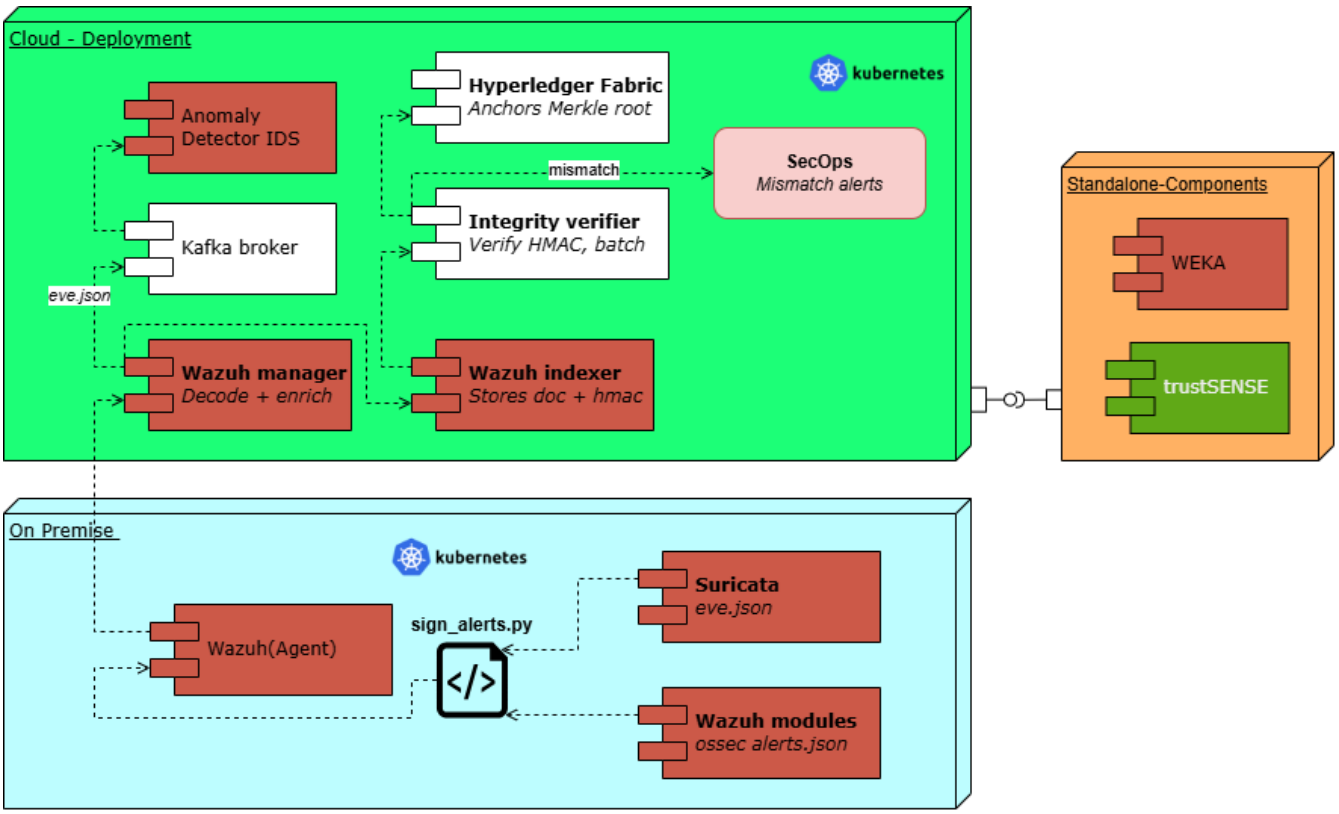


Figure 1. Positioning of D4.3's components in the CYberSynchrony platform

The architecture is organized across three deployment zones: an on-premise layer for local data collection and preparation, a cloud deployment layer running under Kubernetes for processing and analysis, and a set of standalone components operating independently of the core pipeline. The diagram illustrates interdependencies of the documented services with other project's components too, which are documented in other deliverables.

© CYberSynchorny, 2026

3 Suricata IDS

3.1 Updates since V1

Suricata is deployed in CYberSynchrony as the network intrusion detection component responsible for monitoring network traffic in the on-premise environment. Its role is to inspect packets, identify suspicious network behaviour, detect traffic matching known IDS rules, and produce structured network security events. Within the project architecture, Suricata observes the actual network traffic and produces detailed technical information, including source and destination IPs, ports, protocols, application protocols, alert signatures, HTTP hostnames, URLs, flow packet counts, byte counts, and payload-related fields.

Note: Suricata is an additional network monitoring/IDS component that was not explicitly foreseen in Section 2.1.4 in the Grant Agreement but was identified as necessary to strengthen network data and provide valuable traffic data to other tools (e.g. DCO Response Playbooks, Anomaly Detector). This tool offers increased added value for CyberSynchrony which was not foreseen, as Suricata improves threat detection, enriches the unified alert stream, and provides stronger evidence for network traffic packet analytics and integrity validation.

3.2 Integration, interoperability and deployment

The Suricata deployment follows the CYberSynchrony hybrid deployment model, which includes two Kubernetes environments provided by eBOS: one cloud cluster for computationally heavier processes and one on-premise cluster for local monitoring and telemetry collection. Suricata is deployed on the on-premise side, where it inspects real network traffic and generates structured security events in eve.json format. These events are collected by the Wazuh agent, sent to the cloud-side Wazuh Manager, visualised in the Wazuh Dashboard, and made available for processing by other platform components, including MISP, the Anomaly Detector and DCO Response Playbooks. The architecture can be summarized as follows:

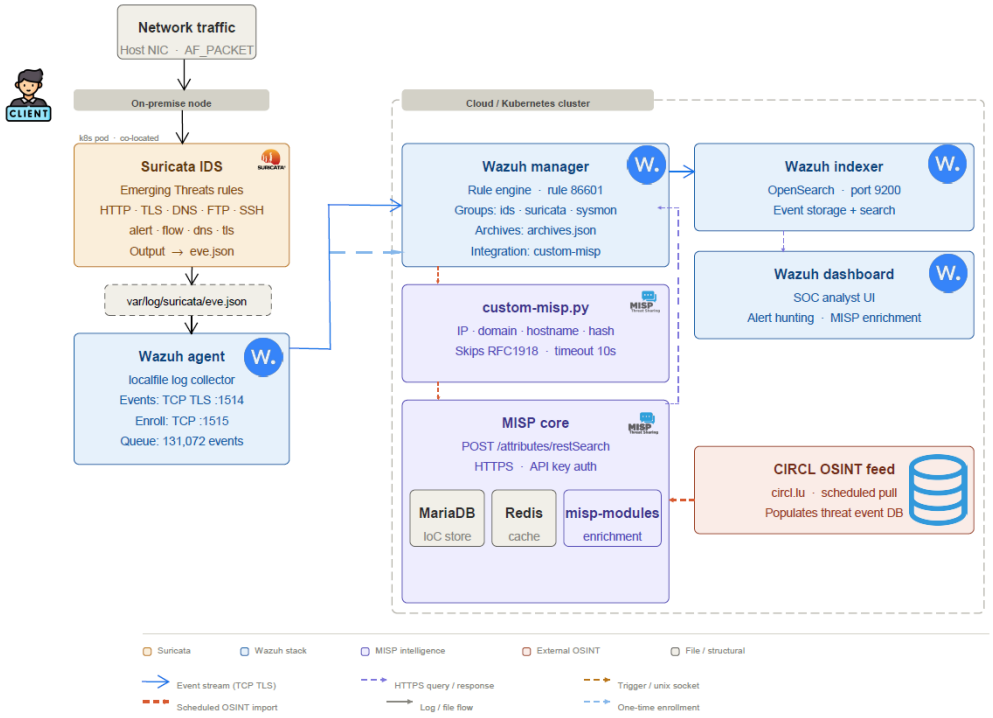


Figure 2. Wazuh-Suricata-MISP integration and data flow

© CYberSynchorny, 2026

3.2.1 Deployment Configuration

Suricata is deployed on the on-premise side of the CYberSynchrony environment. The most important deployment decision was enabling Suricata to inspect the correct interface. Initially, the Suricata start script used eth0, which inside a Kubernetes pod refers to the pod's internal virtual interface. This meant Suricata could not see the host-level traffic needed for monitoring. The corrected deployment uses **hostNetwork: true**, **dnsPolicy: ClusterFirstWithHostNet**, and **SURICATA_INTERFACE=ens34**, allowing Suricata to bind to the real host interface. Furthermore, the deployment was corrected to address three main issues: incorrect HOME_NET, incorrect network interface selection, and insufficient output configuration. The relevant files (*on the project's GITLAB repository*) changed were:

Table 2: Suricata Deployment issues

File	Purpose
k8s/suricata/configmap.yaml	Suricata configuration, including HOME_NET, protocol parsers and outputs.
k8s/suricata/deployment.yaml	Kubernetes deployment settings, including host networking and interface configuration.
suricata/start.sh	Runtime interface selection and Suricata start command.

The first correction concerned **HOME_NET**. The original configuration only included:

```
HOME_NET: "[192.168.0.0/16]"
```

This was not sufficient for the CYberSynchrony environment, because the actual network includes Tailscale and private address ranges:

- 100.64.0.0/10
- 10.0.0.0/8
- 172.16.0.0/12
- 192.168.0.0/16

Suricata uses **HOME_NET** to determine whether a network is internal or external. Many IDS rules depend on this distinction. If **HOME_NET** is incorrect, Suricata may see traffic but fail to trigger the relevant rules, because it does not correctly classify the monitored environment. The fix expanded **HOME_NET** to cover the actual project network ranges and populated related server groups, such as HTTP, SMTP and DNS.

The second correction concerned the network interface. The previous startup command was:

```
suricata -c /etc/suricata/suricata.yaml -i eth0 &
```

Inside the Kubernetes environment, eth0 is the pod-internal virtual interface. It does not necessarily see the host's physical network traffic. The corrected deployment uses host networking and dynamic interface selection:

```
SURICATA_IFACE="${SURICATA_INTERFACE:-$(ip route show default | awk '/default/
{print $5; exit}')}"

SURICATA_IFACE="${SURICATA_IFACE:-eth0}"

suricata -c /etc/suricata/suricata.yaml -i "$SURICATA_IFACE" &
```

In the current deployment, the relevant interface is ens34. This allows Suricata to monitor the real host network traffic instead of only the pod's internal traffic.

© CYberSynchorny, 2026

The third correction concerned Suricata output configuration. The previous outputs were too narrow and mainly included alert, flow and stats. The corrected configuration enables richer protocol-level output, including HTTP, DNS, TLS, SSH, file metadata, anomaly detection and payload-related fields. This is important because downstream tools require more than a basic alert. They need connection data, protocol data, traffic volume and contextual fields for correlation, AI detection and response workflows.

3.2.2 Indicative example

Suricata produces raw network security events in the eve.json format. These are structured JSON records that contain the details of the detected network event. A representative raw Suricata event is shown below:


```
{
  "timestamp": "2026-04-23T09:40:50.013486+0000",
  "flow_id": "2003374226620530.000000",
  "in_iface": "ens34",
  "event_type": "alert",
  "src_ip": "143.204.194.100",
  "src_port": "80",
  "dest_ip": "100.64.12.14",
  "dest_port": "44532",
  "proto": "TCP",
  "alert": {
    "signature_id": "2100498",
    "signature": "GPL ATTACK_RESPONSE id check returned root",
    "category": "Potentially Bad Traffic",
    "severity": "2"
  },
  "http": {
    "hostname": "testmynids.org",
    "url": "/uid/index.html",
    "http_method": "GET"
  },
  "flow": {
    "pkts_toserver": "6",
    "bytes_toserver": "496"
  }
}
```


This raw event contains the key information required for network-level security analysis: when the event occured, which interface observed it, the source and destination of the connection, the protocol, the Suricata rule that triggered it, the HTTP hostname and URL, and basic flow volume data. The fields of this event are explained below:

Table 3: Suricata Fields

Field	Meaning
timestamp	Time when Suricata observed the event.
flow_id	Identifier of the network flow.
in_iface	Network interface where the event was observed, currently ens34.

© CYberSynchorny, 2026

event_type	Type of Suricata event, for example alert.
src_ip, src_port	Source IP and port.
dest_ip, dest_port	Destination IP and port.
proto	Transport protocol, for example TCP or UDP.
app_proto	Application protocol, for example HTTP, DNS or TLS, when detected.
alert.signature_id	Suricata rule identifier.
alert.signature	Human-readable Suricata alert signature.
alert.category	Category of the detected activity.
alert.severity	Suricata severity score.
http.hostname	HTTP hostname, when the event is HTTP-related.
http.url	HTTP URL, when available.
flow.pkts_toserver	Packets sent from client to server.
flow.pkts_toclient	Packets sent from server to client.
flow.bytes_toserver	Bytes sent from client to server.
flow.bytes_toclient	Bytes sent from server to client.
flow.start	Flow start time, used to derive traffic rates.
payload, payload_printable, packet	Packet/payload data, when enabled and available.

3.2.3 Integration with Wazuh

Suricata is integrated with Wazuh via the Wazuh agent and the Wazuh Manager pipeline, within the on-premise cluster. The raw Suricata event is first produced in eve.json on the on-premise side. The Wazuh agent collects it and forwards it to the cloud-side Wazuh Manager. The Wazuh Manager then decodes the JSON event, applies rule matching, and stores the resulting alert in the unified Wazuh alert file (see Figure 2). The Wazuh rule currently used for Suricata alerts is:

```
86601 - Suricata: Alert - $(alert.signature)
```

This rule catches Suricata events where event_type is alert and classifies them with:

```
"groups": ["ids", "suricata"]
```

This means that once Suricata events are processed by Wazuh, they can be filtered in the Wazuh Dashboard using rule.groups: suricata or rule.groups: ids.

© CYberSynchorny, 2026

Before Suricata was added, the Wazuh unified alert file contained only Wazuh-native events. These included SCA compliance checks, rootcheck events and agent lifecycle events. After Suricata integration, IDS events from Suricata appeared in the same file and in the same Wazuh Dashboard.

Table 4: Before and After Suricata integration

Field	Meaning
Before Suricata integration	Wazuh-native events only, such as SCA checks, rootcheck alerts and agent lifecycle events.
After Suricata integration	Wazuh-native events plus Suricata IDS alerts, including network traffic detections, signatures, source/destination IPs, HTTP fields and flow data.

This is very important, because it demonstrates that Wazuh can act as a unified security monitoring layer for both host- and network-based events. The dashboard validation also confirmed this change (see next section for Wazuh Dashboard screenshots). With a filter excluding Suricata, the dashboard showed only Wazuh-native events. Without that filter, Suricata alerts appeared alongside Wazuh events. The Wazuh Indexer stores both event types under the same daily index pattern, as there is no separate suricata-* index. The source is distinguished by the rule.groups field inside each document.

```
wazuh-alerts-4.x-YYYY.MM.DD
```

3.2.4 Communication with other platform components

Suricata data is consumed from other platform components too, which are described below:

Table 5: Suricata communication with other components

Platform component	Partner	Usage
MISP	IANUS	Uses indicators extracted from Suricata/Wazuh alerts (e.g. IP addresses, domains, URLs and hashes) and checks them against MISP IoC repositories and external threat intelligence feeds. When a match is found, MISP enriches the original alert with threat intelligence context (e.g. IoC type, related event, category, threat level and source organisation) and the enriched event is sent back to Wazuh Server and visible in the Wazuh Dashboard.
Kafka	MAG	Streams Suricata/Wazuh events from the Wazuh Server logs into Kafka topics so that other CYberSynchrony components can consume the same monitoring data in near real time. This avoids each component needing a separate direct connection to Suricata and supports scalable data distribution across the platform.
Blockchain Integrity Validator	BYTE	The Blockchain Integrity Validator receives verified Wazuh/Suricata alert records and stores cryptographic integrity evidence for them. Alerts are signed or hashed before or during ingestion, the verifier recomputes the HMAC/hash to confirm that the event has not been modified, and verified alerts are grouped into batches whose Merkle root is written to Hyperledger. This allows later forensic checks to

© CYberSynchorny, 2026

		prove that the Wazuh/Suricata alert data used for investigation, reporting or anomaly detection has not been tampered.
Anomaly Detector IDS	MAG	Uses the network traffic and flow-related fields collected by Kafka (which is subscribed to Wazuh Server), such as source/destination IPs, ports, protocol, packet counts, byte counts and flow timing. These features are used to train and operate the AI-based Anomaly Detection System so it can identify unusual network behaviour beyond signature-based IDS alerts, based on the network traffic packets and telemetry data gathered by Suricata.
DCO Response Playbooks	FP	Uses the aggregated alert data produced by Wazuh and Suricata and MISP. Based on this aggregated context, the DCO component can provide response playbooks or recommendations to support analysts in deciding the appropriate mitigation or escalation steps.
Wazuh Server / Dashboard	IANUS	Provides the operational interface where analysts can view, search and filter Suricata alerts together with Wazuh-native events and MISP-enriched threat intelligence alerts. This gives SOC analysts a single dashboard for reviewing network IDS detections, endpoint events, vulnerability information and enriched threat intelligence context.

A representative sample - extracted from the Wazuh Server database - is as follows:


```
{
  "timestamp": "2026-04-23T09:40:51.339+0000",
  "agent": "cybersync4",
  "src_ip": "143.204.194.100",
  "src_port": "80",
  "dest_ip": "100.64.12.14",
  "dest_port": "44532",
  "proto": "TCP",
  "app_proto": "http",
  "signature_id": "2100498",
  "signature": "GPL ATTACK_RESPONSE id check returned root",
  "severity": "2",
  "flow_pkts_toserver": "6",
  "flow_pkts_toclient": "5",
  "flow_bytes_toserver": "496",
  "flow_bytes_toclient": "876",
  "flow_start": "2026-04-23T09:40:49.868466+0000",
  "http_hostname": "testmynids.org",
  "http_url": "/uid/index.html",
  "http_method": "GET",
  "http_user_agent": "curl/7.81.0"
}
```


In the current deployment, Suricata is integrated via structured event output rather than a dedicated REST API ecosystem. Its main operational interface is the eve.json event stream, which Wazuh collects and transforms into Wazuh alerts. Platform components can then consume Wazuh & Suricata information through the Wazuh

© CYberSynchorny, 2026

Server, the Wazuh Dashboard, Kafka-based streaming, or dedicated integration endpoints built on top of the aggregated Wazuh/Suricata data.

© CYberSynchorny, 2026

4 Wazuh

4.1 Updates since V1

Wazuh is deployed in CYberSynchrony as the central security monitoring, alert normalisation and dashboard component. Its role is to receive events from the monitored on-premise environment, including Wazuh-native endpoint events and Suricata IDS alerts, process them through the cloud-side Wazuh Manager, store them in the Wazuh Indexer, and expose them to analysts through the Wazuh Dashboard. The main update since v1 is the successful integration of Wazuh with Suricata and MISP as part of the wider CYberSynchrony monitoring pipeline. Wazuh now acts as the common operational layer where host-based alerts, network IDS events and threat-intelligence-enriched events can be reviewed together. It also produces a unified alert stream that can be reused by other platform components, including Kafka-based data flows, the blockchain integrity layer, MAG's Anomaly Detector IDS and FP's DCO Response Playbooks. In this architecture, Wazuh is not used only as a standalone SIEM tool. Its main value for CYberSynchrony is that it converts different monitoring inputs into a common alert structure. This allows the platform to use Wazuh as a shared source of operational security evidence for detection, enrichment, integrity validation, analytics and response recommendation workflows.

4.2 Integration, interoperability and deployment

The Wazuh deployment follows the CYberSynchrony hybrid architecture. The monitored on-premise environment includes the Wazuh agent and Suricata. The Wazuh agent (on-premise) forwards the alert events to the cloud-side Wazuh Manager, where they are decoded, matched against Wazuh rules, enriched with metadata and written into the unified alert stream. The Wazuh Indexer stores the resulting alerts in searchable indices, while the Wazuh Dashboard provides the user-facing interface for analysts.

Table 6: Wazuh Components

Wazuh Component	Deployment	Role in architecture
Wazuh Manager	Cloud Kubernetes	Receives events from agents, decodes logs, applies rules, enriches alerts and writes the unified alert file.
Wazuh Indexer	Cloud Kubernetes	Stores Wazuh alerts in searchable indices and supports dashboard queries.
Wazuh Dashboard	Cloud Kubernetes	Provides the user interface for searching, filtering and visualising alerts.
Wazuh Agent	On-premise Kubernetes	Collects local endpoint events and Suricata eve.json events from the on-premise environment and forwards them to the Wazuh Manager.

The Wazuh Manager writes alerts into a unified JSON Lines file. This file is located inside the Wazuh Manager pod at:

```
/var/ossec/logs/alerts/alerts.json
```

The same file is also available on the cloud host through the hostPath-mounted location:

```
/data/wazuh-manager/logs/alerts/alerts.json
```

This file is the main operational output of the Wazuh deployment. It contains Wazuh-native alerts, such as SCA compliance checks, vulnerability-related findings, rootcheck events and agent lifecycle events, as well as Suricata-derived IDS alerts. Each alert is written as one complete JSON object, making the stream suitable for

© CYberSynchorny, 2026

dashboard visualisation, hashing, Kafka streaming, MISP correlation and dataset generation for the Anomaly Detector IDS.

4.2.1 Accessing the Manager pod

The Wazuh Manager pod can be accessed through Kubernetes. The following commands were used to access the manager and inspect the unified alert file:

```
ssh ianus@100.64.12.11

sudo kubectl -n ianus-lab exec -it $(sudo kubectl -n ianus-lab get pod -l
app=wazuh-manager -o name | head -1) -- bash

cd /var/ossec/logs/alerts

tail -n 20 alerts.json
```

The unified alert file is append-only. New alerts are added at the end of the file. This is important because downstream services can safely read the file continuously without interfering with Wazuh's operation. Platform components can consume the file using standard tools such as tail, grep, jq, Fluent Bit, Filebeat, Logstash, Kafka Connect FileStreamSource or any programming language with a JSON parser. Every alert line in alerts.json is a complete JSON object. The structure can be understood as two layers.

- The first layer is the Wazuh outer shell. This is added by the Wazuh Manager during processing. It includes information such as timestamp, rule, agent, manager, decoder, alert ID and source location.
- The second layer is the data object. This preserves the original source event. For Suricata alerts, this means the original fields from Suricata's eve.json are preserved under data. For Wazuh-native events, such as SCA checks or rootcheck findings, the data object contains the corresponding Wazuh security assessment or anomaly information. The main outer fields are:

Table 7: Main fields of the Wazuh alert structure

Field	Explanation
timestamp	Time when Wazuh processed the event.
rule	Wazuh rule that matched the event.
agent	Agent that produced or forwarded the event.
manager	Wazuh Manager pod that processed it.
id	Internal Wazuh alert ID.
decoder	Decoder used to parse the raw event.
data	Preserved source-specific event data.
location	Original source location, for example Suricata eve.json, SCA or rootcheck.
full_log	Raw log line, when available.

© CYberSynchorny, 2026

Note: The rule object is especially important because it is the main Wazuh enrichment layer. It provides severity, description, rule ID and categorisation groups. Depending on the matched rule, it may also include compliance mappings such as PCI DSS, GDPR, HIPAA, NIST, ISO 27001, CIS, MITRE, SOC 2 or other mappings.

Table 8: Main fields of the Wazuh rule object

Field	Explanation
ule.level	Wazuh severity level from 0 to 15. Higher means more critical.
rule.description	Human-readable explanation of the alert.
rule.id	Wazuh rule identifier.
rule.groups	Alert categories, for example ids, suricata, sca, ossec, rootcheck.
rule.pci_dss, rule.gdpr, rule.hipaa	Compliance mappings when available for the specific rule.
rule.mitre_tactics, rule.mitre_techniques	MITRE ATT&CK mappings when available.

For example, Suricata alerts are currently matched by Wazuh rule 86601, which identifies them and assigns them to the groups ["ids", "suricata"]. Wazuh-native SCA events, on the other hand, can include richer compliance mappings, such as PCI DSS, NIST, HIPAA, ISO 27001, CIS and MITRE mappings. This shows that the enrichment level depends on the Wazuh rule that matched the event.

The following examples illustrate how different event types are represented after processing by Wazuh. They show the common structure used for both Suricata-derived network alerts and Wazuh-native host/security posture events, including the Wazuh rule metadata, agent information, decoder information, source location and preserved event data. These examples are included to demonstrate how Wazuh normalises heterogeneous inputs into a unified alert format that can be visualised in the dashboard and reused by other CYberSynchrony components.

4.2.2 Suricata IDS Alert example

A Suricata IDS alert is generated when network traffic matches a Suricata rule. Wazuh receives the event, wraps it into its own alert structure, and stores it in alerts.json.


```
{
  "timestamp": "2026-04-23T09:40:51.339+0000",
  "rule": {
    "level": 3,
    "description": "Suricata: Alert - GPL ATTACK_RESPONSE id check returned
root",
    "id": "86601",
    "groups": ["ids", "suricata"]
  },
  "agent": {
    "id": "001",
```


© CYberSynchorny, 2026

```
    "name": "cybersync4",
    "ip": "100.64.12.14"
  },
  "manager": {
    "name": "wazuh-manager-6bd75cd958-qsrrq"
  },
  "decoder": {
    "name": "json"
  },
  "data": {
    "src_ip": "143.204.194.100",
    "dest_ip": "100.64.12.14",
    "proto": "TCP",
    "alert": {
      "signature_id": "2100498",
      "signature": "GPL ATTACK_RESPONSE id check returned root"
    },
    "http": {
      "hostname": "testmynids.org"
    },
    "flow": {
      "pkts_toserver": "6",
      "bytes_toserver": "496"
    }
  },
  "location": "/var/log/suricata/eve.json"
}
```


This alert is important because it shows the interoperability between Suricata and Wazuh. Suricata provides the raw network detection, while Wazuh adds rule metadata, agent information, manager information, decoder information and unified storage.

4.2.3 Wazuh SCA Compliance Alert

A Wazuh also produces native SCA alerts. These relate to security configuration assessment and compliance checks. For example, Wazuh can detect that a specific CIS benchmark control failed and then attach compliance and MITRE mappings to that event.


```
{
  "timestamp": "2026-04-22T12:23:54.415+0000",
  "rule": {
    "level": 7,
    "description": "CIS Ubuntu Linux 22.04 LTS Benchmark v2.0.0.: Ensure
pam_pwhistory includes use_authtok.",
    "id": "19007",
    "groups": ["sca"],
    "pci_dss": ["2.2"],
    "nist_800_53": ["CM.1"],
    "hipaa": ["164.312(a)(2)(iv)"],
    "mitre_tactics": ["TA0006"]
  },
```


© CYberSynchorny, 2026

```
  "agent": {
    "id": "001",
    "name": "cybersync4",
    "ip": "100.64.12.14"
  },
  "decoder": {
    "name": "sca"
  },
  "data": {
    "sca": {
      "type": "check",
      "policy": "CIS Ubuntu Linux 22.04 LTS Benchmark v2.0.0.",
      "check": {
        "title": "Ensure pam_pwhistory includes use_authtok.",
        "result": "failed"
      }
    }
  },
  "location": "sca"
}
```


This example demonstrates that Wazuh is not only used for event monitoring, but also for compliance and security posture information. This is directly useful for CYberSynchrony because compliance-related metadata can be used by downstream analytics, dashboards, reporting functions and policy-alignment tools.

4.2.4 Wazuh Rootcheck Alert

Rootcheck events are Wazuh-native anomaly findings. They identify suspicious or unusual local system conditions, such as unexpected files or possible hidden resources.


```
{
  "timestamp": "2026-04-23T07:40:37.043+0000",
  "rule": {
    "level": 7,
    "description": "Host-based anomaly detection event (rootcheck).",
    "id": "510",
    "groups": ["ossec", "rootcheck"],
    "pci_dss": ["10.6.1"]
  },
  "agent": {
    "id": "001",
    "name": "cybersync4"
  },
  "manager": {
    "name": "wazuh-manager-6bd75cd958-qsrrq"
  },
  "full_log": "File '/dev/termination-log' present on /dev. Possible hidden
file.",
  "decoder": {
    "name": "rootcheck"
  },
  "data": {
```


© CYberSynchorny, 2026

```
    "title": "File present on /dev.",
    "file": "/dev/termination-log"
  },
  "location": "rootcheck"
}
```


This shows that the same Wazuh deployment supports both network-driven events coming from Suricata and host-based security events produced by Wazuh itself.

4.2.5 Communication with other platform components

Table 9: Wazuh communication with other tools

Consumer	Wazuh usage
Blockchain hashing component	Hashes alert records for integrity validation.
Kafka consumer	Suricata network events are streamed to the Kafka topic `on-premise-info` via a Filebeat instance running in the Suricata container. Wazuh alerts are indexed by the Wazuh Indexer and forwarded to MISP via the Wazuh active integration.
MISP correlation	Extracts IoCs such as IPs, domains, URLs and hashes for threat intelligence lookup.
MAG Anomaly Detector pipeline	Uses traffic volume, connection and rate features from Kafka consumer.
DCO Response Playbooks	Can use alert severity, type and context for response recommendation workflows.

The difference between raw input and Wazuh-processed output is important. Before Wazuh processing, Suricata produces a raw eve.json network event. After Wazuh processing, the same event is transformed into a structured Wazuh alert that includes rule metadata, agent information, manager traceability, decoder information and source location.

Table 10: Before and after integration

Stage	Description
Before Wazuh processing	Raw Suricata eve.json event with network fields such as source IP, destination IP, protocol, signature and flow information.
After Wazuh processing	Wazuh alert in alerts.json, containing the original Suricata data plus Wazuh rule, severity, agent, manager, decoder and location metadata.

Overall, the Wazuh deployment has been completed, integrated and validated as the central monitoring and alert normalisation layer for the CYberSynchrony environment. Wazuh receives events from the on-premise environment, integrates Suricata IDS alerts, enriches them with Wazuh rule metadata, stores them in the Wazuh Indexer, exposes them through the Wazuh Dashboard, and writes them to a unified JSON Lines file for downstream consumers. The deployment confirms that Wazuh can support multiple project needs at once:

© CYberSynchorny, 2026

operational monitoring, alert visualisation, compliance-related checks, threat intelligence correlation through MISP, data streaming through Kafka, blockchain hashing, and dataset creation for MAG's Anomaly Detector. The main remaining work concerns infrastructure hardening, ~~MAG ingestion alignment,~~ and Keycloak-based SSO configuration.

4.2.6 Wazuh Dashboard

The Wazuh Indexer stores the alerts produced by the Wazuh Manager. The Wazuh Dashboard queries the Indexer and provides the visual interface for analysts. The deployment validation confirmed that alerts are stored in daily Wazuh indices using the pattern:

```
wazuh-alerts-4.x-YYYY.MM.DD
```

The dashboard can display both Wazuh-native events and Suricata-derived IDS alerts in a single interface. This was validated by comparing dashboard views before and after enabling Suricata visibility. Before Suricata integration, the dashboard showed only Wazuh-native events such as SCA, rootcheck and agent lifecycle events. After the integration, Suricata alerts appeared alongside the Wazuh-native events.

In Figure 3, the Wazuh Dashboard presents a consolidated view of the monitored endpoint, including event evolution, MITRE ATT&CK mapping, compliance status, vulnerability detection results and the latest SCA scans. The dashboard shows the number and severity of detected vulnerabilities, the most affected software packages, compliance mappings such as PCI DSS, and the latest configuration assessment results for the monitored Ubuntu endpoint.

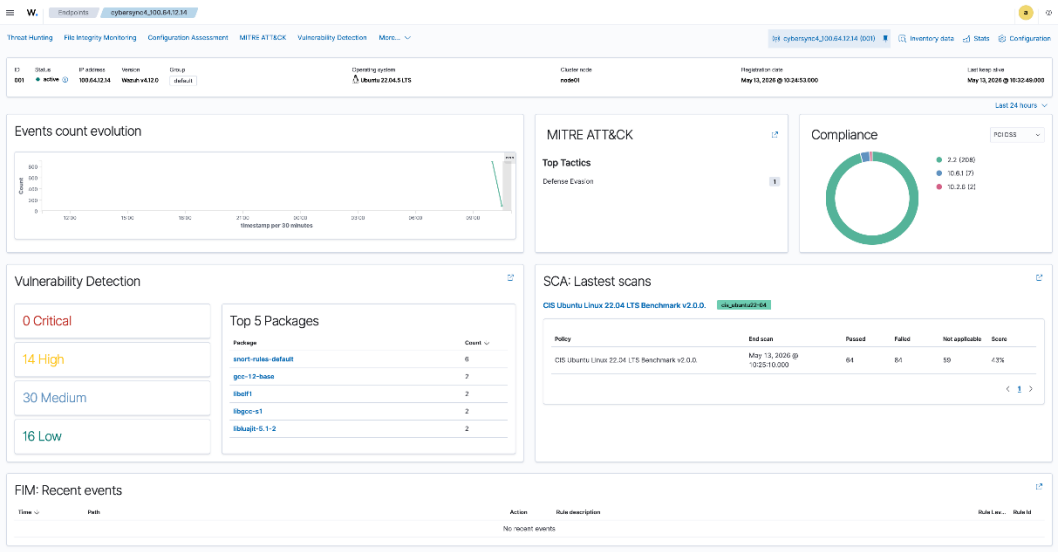


Figure 3. Vulnerability Dashboard in Wazuh

In Figure 4, the Vulnerability Detection dashboard provides a focused view of the vulnerabilities identified on the monitored endpoint, filtered for high-severity findings. The dashboard shows the distribution of vulnerabilities by severity, the top detected CVEs, the affected operating system, the monitored agent, the most vulnerable packages, and the publication year of the identified vulnerabilities.

© CYberSynchorny, 2026

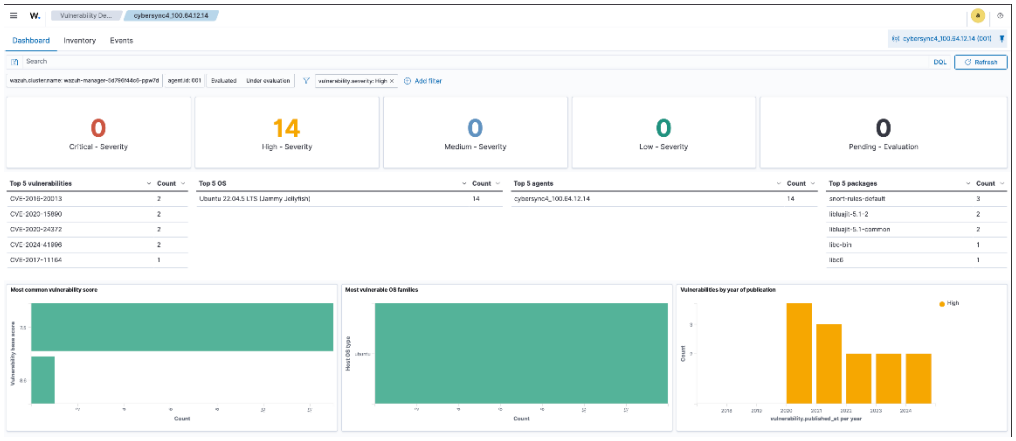


Figure 4. Vulnerabilities discovered in Dashboard

In Figure 5, Figure 6, Figure 7 and Figure 8, the Wazuh Threat Hunting dashboard provides a consolidated view of security events collected from the monitored endpoint during the selected time window. The dashboard shows the total number of alerts, alert evolution over time, top alert groups, rule groups, and PCI DSS mappings, while confirming the presence of Suricata and MISP event categories and validating successful project integration. Native Wazuh alerts, network telemetry alerts from Suricata and threat intelligence data from MISP are successfully integrated into Wazuh and can be reviewed together by SOC analysts. For example, in Figure 8, the visible alerts include SSH login attempts with non-existent users, repeated password mistakes, a PAM login failure, SCA compliance findings for the CIS Ubuntu 22.04 benchmark, and an alert indicating that the monitored interface entered promiscuous/sniffing mode.

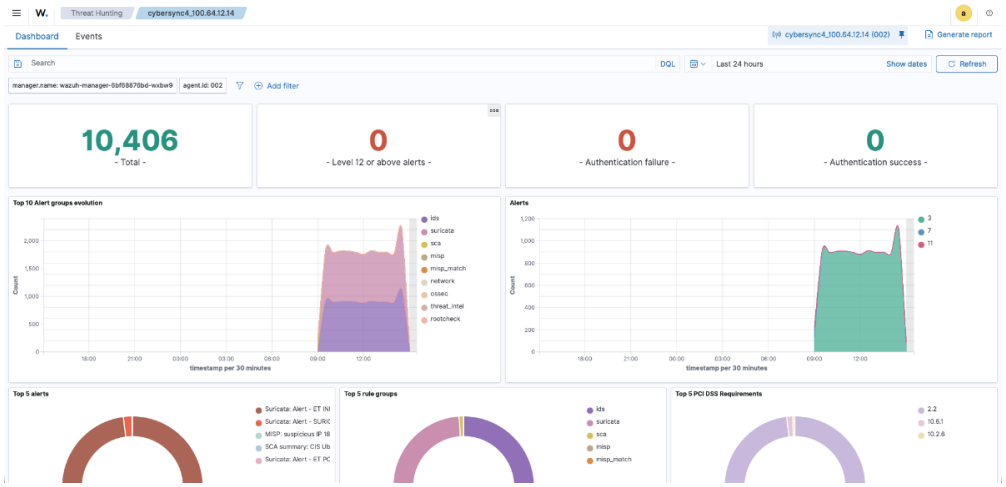


Figure 5. Wazuh Threat Hunting Dashboard

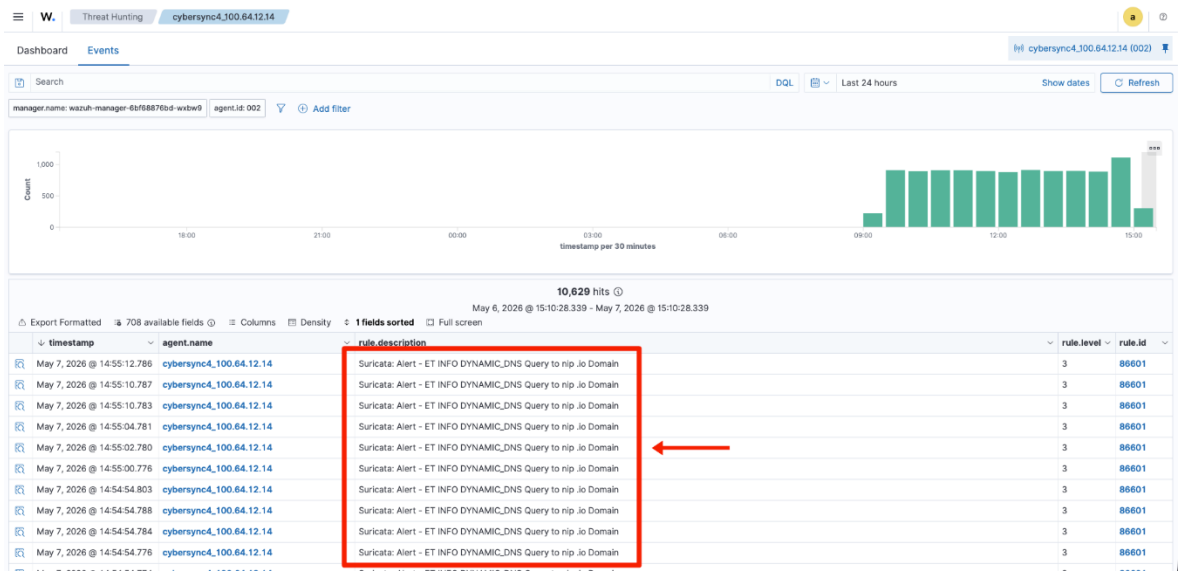


Figure 6. Wazuh Threat Hunting Dashboard (Suricata events)

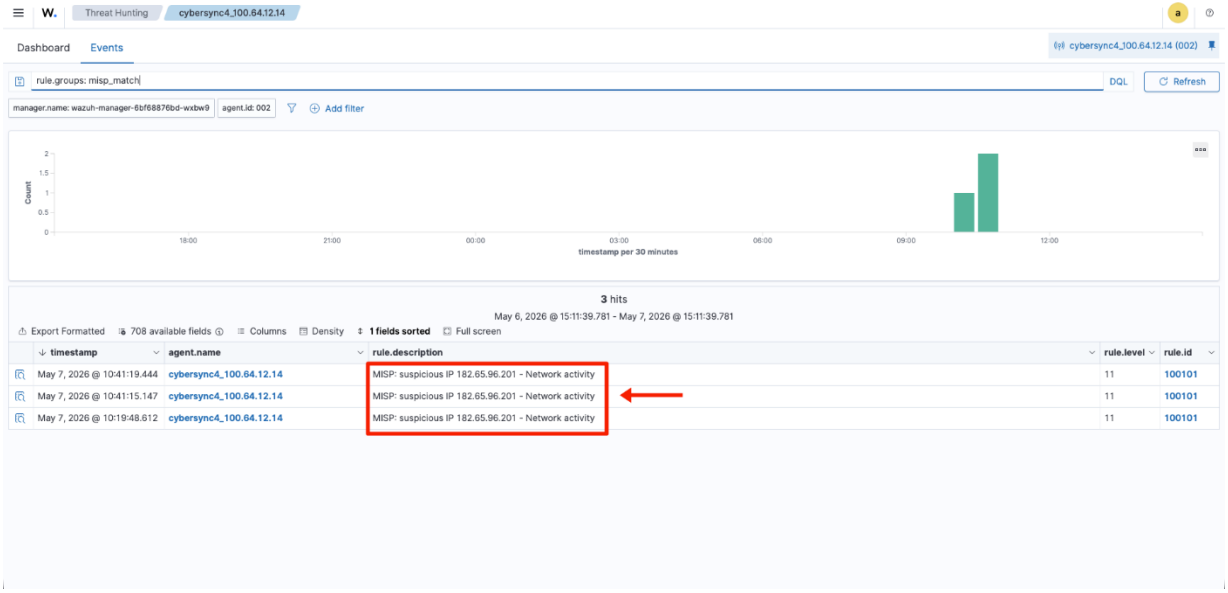


Figure 7. Wazuh Threat Hunting Dashboard (MISP events)

© CYberSynchorny, 2026

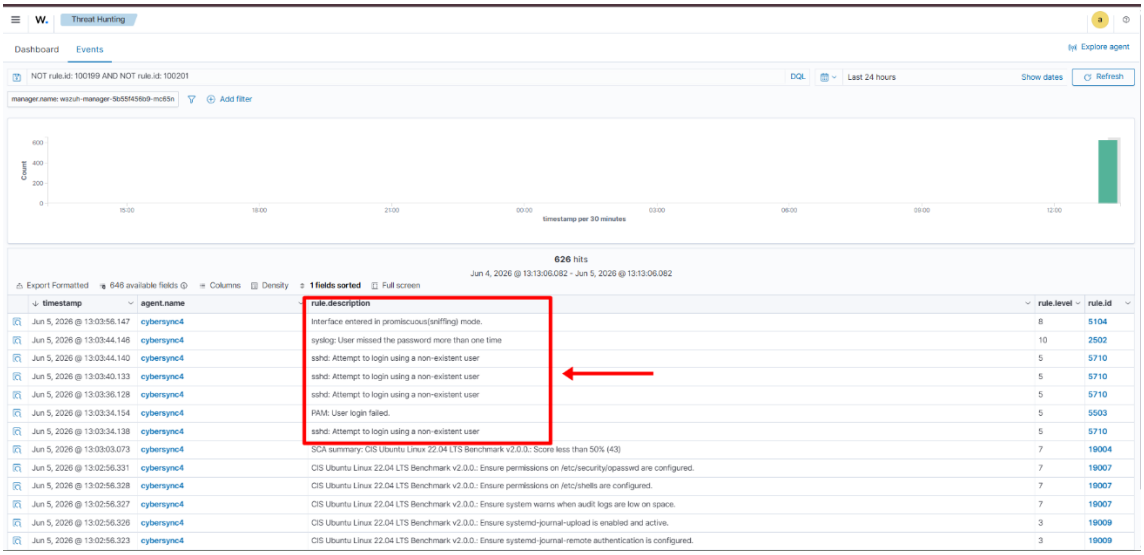


Figure 8. Wazuh Threat Hunting Dashboard (Wazuh events)

As for the Malware Detection offered by Wazuh, in Figure 9 and Figure 10, the Wazuh Malware Detection dashboard shows host-based anomaly findings generated through Wazuh rootcheck rules. The dashboard identifies events such as files present under /dev and files owned by root with write permissions for anyone, while presenting the affected agent, event timeline, rule group, rule description, severity level and rule ID. The added value is that Wazuh not only visualise Suricata network alerts, but also provides host-level monitoring and anomaly detection from the same interface. This allows analysts to correlate endpoint anomalies, malware-related indicators, network IDS detections and enriched threat intelligence within a unified monitoring environment.

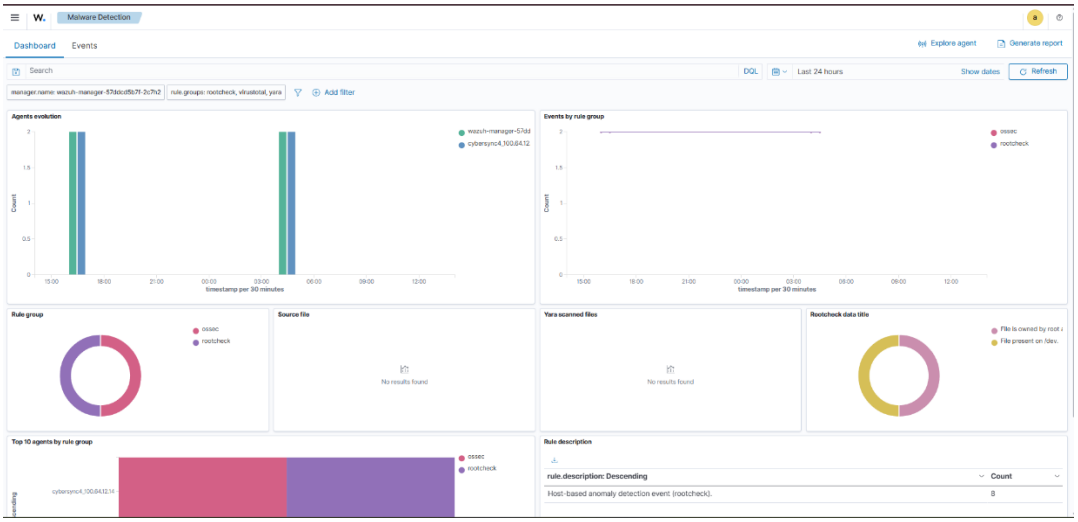


Figure 9. Wazuh Malware Detection Dashboard

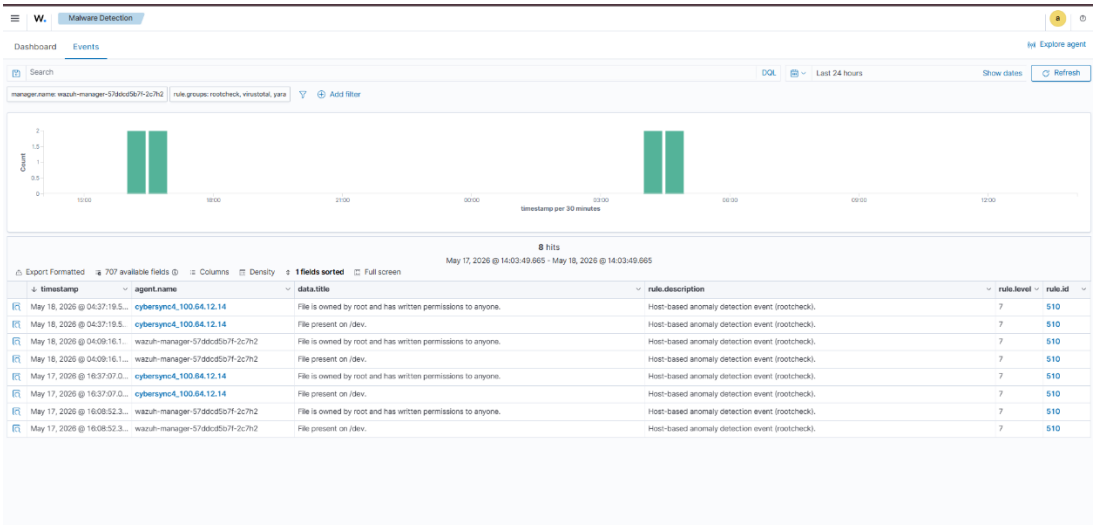


Figure 10. Wazuh Malware Detection Events

When an event is selected in the Wazuh Malware Detection dashboard, a document details panel opens on the right-hand side of the interface. This panel provides the full structured information for the selected event, including the index, agent ID, agent IP, agent name, affected file, event title, decoder, full log message, manager name, rule description, rule groups, rule ID, severity level, compliance mappings and timestamp. In the example shown, the selected event is a Wazuh rootcheck finding with rule ID 510 and severity level 7. The message indicates that the file /dev/termination-log is owned by root and has write permissions for anyone, which is reported as a host-based anomaly detection event. This detailed view allows analysts to move from the dashboard summary to the exact technical evidence behind the alert, supporting investigation, validation and reporting.

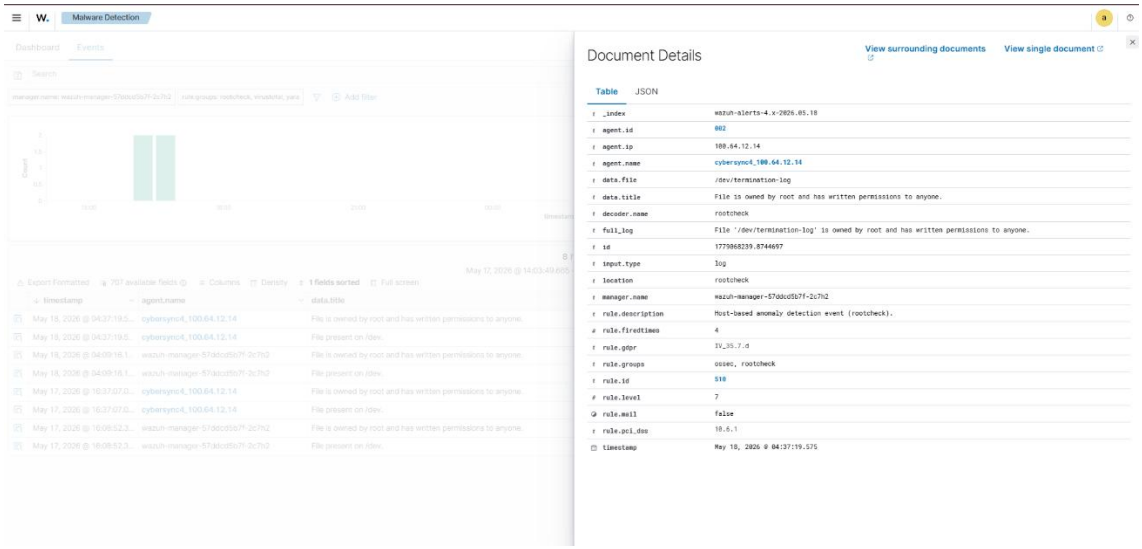


Figure 11. Wazuh Malware Detection Event details

5 Anomaly detector IDS

5.1 Updates since V1

Since the first version of the Integrated Cybersecurity Monitoring Suite, the Anomaly Detector IDS has been further consolidated to better support the integrated CYberSynchrony deployment pipeline. The main updates concern the operational input source, the inference logic and the event exchange mechanism with the rest of the monitoring stack.

The component has been aligned with the Suricata-based telemetry flow. In the current configuration, the detector processes Suricata `eve.json` outputs, with emphasis on flow-level features. This allows the component to consume structured network events already produced by the CYBRITE network monitoring layer, without requiring a separate traffic capture mechanism.

The detection model has been refined around a transformer-based autoencoder. The encoder applies the MultiHead Self-Attention mechanism, documented in D4.1, to capture relationships across the feature dimensions before compressing the input to a compact bottleneck representation. The decoder reconstructs the original vector from that representation. The model is trained exclusively on benign network flow records, with no labelled attack data required; architecture and training hyperparameters were selected through an automated search using the optimization library Optuna[1]. Training was conducted on a corpus of normal network behaviour collected over a two-week observation period from the target environment.

The preprocessor converts each incoming flow record into a 34-dimensional feature vector. The vector is composed of three groups. The first covers thirteen numerical measurements derived from the flow block: packet counts, byte counts, per-direction throughput, mean bytes per packet, flow duration, and packet rate. The second adds two ratio features expressing directional asymmetry in packet and byte counts, bounded between −1 and 1. The third encodes transport protocol, application protocol, and flow state as one-hot vectors across fixed vocabularies, producing nineteen binary dimensions. Values outside the known vocabulary for any categorical field are mapped to an explicit unknown category rather than dropped. The resulting vector has a fixed schema. Any change to the feature definition requires a coordinated update of both the preprocessor and the detection model, followed by a full retraining cycle.

At inference time, the model computes a reconstruction error for each flow vector — the mean squared error between the normalised input and its reconstruction:

$$score(x) = mean[(\, norm(x) - model(x)\,)^2]$$

A flow is flagged as anomalous if this score exceeds a pre-calibrated threshold derived from the distribution of reconstruction errors on a held-out benign validation set, stored with the model artefact and requiring no manual calibration at deployment.

Evaluation of detection performance against simulated attack scenarios is planned as part of the forthcoming pilot activities.

5.2 Integration, interoperability and deployment

5.2.1 Role within the CYBRITE monitoring chain

The AD IDS is positioned after the network telemetry generation layer. Suricata monitors network traffic and produces structured `eve.json` events. AD consumes the relevant flow events, applies AI-based analysis and produces an anomaly decision. In this configuration, AD complements Suricata, which remains responsible for

[1] https://doi.org/10.48550/arXiv.1907.10902

© CYberSynchorny, 2026

traffic inspection, rule matching and generation of structured network events, while AD focuses on behavioral deviation analysis over the flow-level data produced by Suricata.

Table 11. Functional roles within the CYBRITE monitoring chain

Function	Description
Network telemetry source	Suricata generates structured events from monitored traffic
Event forwarding	Filebeat tails Suricata output and publishes events to Kafka
AI-based analysis	The Anomaly Detector IDS consumes Kafka events and analyses flow-level features
Decision logic	The autoencoder calculates reconstruction error and applies a threshold-based decision
User's functions	Results and statistics are presented in the tool's dashboard

5.2.2 Event ingestion via Kafka

The Anomaly Detector receives network flow data through the CYberSynchrony Kafka broker, which serves as the central message bus for the platform. Suricata EVE JSON records are published to the `on-premise-info` topic as they are generated, without buffering or batching at the source. From there, the AD pipeline picks up the messages asynchronously: a preprocessor service reads from `on-premise-info`, extracts the relevant features, and forwards the structured feature vectors to the `preprocessed-data` topic. The inference service then consumes those vectors and publishes its scoring decisions to the `alerts` topic. Figure 12 illustrates this flow.

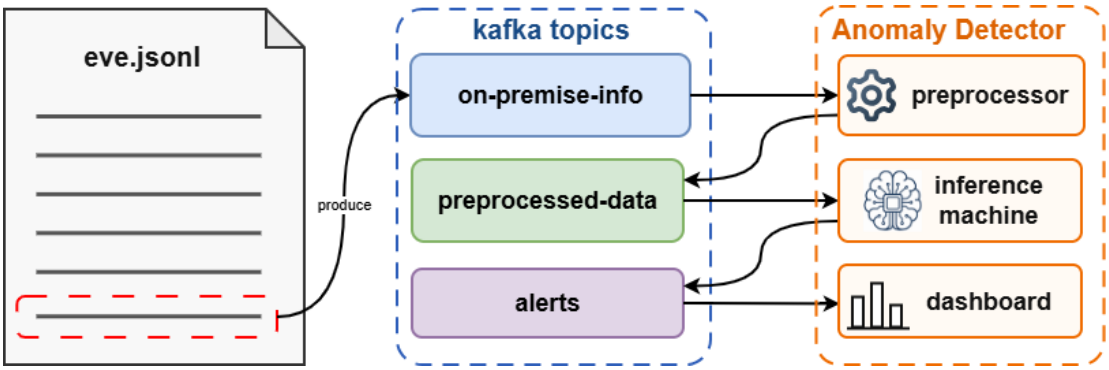


Figure 12. Kafka broker role in Inference pipeline

Of the multiple event types produced by Suricata (alerts, DNS queries, HTTP transactions, TLS handshakes, and others) only records with `event_type: flow` are consumed by the Anomaly Detector. Non-flow records arriving on the `on-premise-info` topic are discarded by the preprocessor. A representative flow record is shown below:


```
{
  "timestamp": "2026-04-27T11:02:06.223256+0000",
  "flow_id": 52526228384558,
  "in_iface": "ens34",
  "event_type": "flow",
  "src_ip": "100.64.12.14",
```

```
  "src_port": 43867,
  "dest_ip": "172.65.247.105",
  "dest_port": 443,
  "proto": "TCP",
  "app_proto": "tls",
  "flow": {
    "pkts_toserver": 348,
    "pkts_toclient": 401,
    "bytes_toserver": 31993,
    "bytes_toclient": 35865,
    "start": "2026-04-27T10:15:11.142126+0000",
    "end": "2026-04-27T11:01:24.872300+0000",
    "age": 2773,
    "state": "closed",
    "reason": "timeout",
    "alerted": false
  },
  "tcp": {
    "tcp_flags": "1f",
    "tcp_flags_ts": "1f",
    "tcp_flags_tc": "1b",
    "syn": true,
    "fin": true,
    "rst": true,
    "psh": true,
    "ack": true,
    "state": "closed"
  }
}
```


5.2.3 Runtime deployment

The Anomaly Detector is deployed as a set of containerised services, each packaged as a Docker image and stored in the project's container registry. The pipeline consists of three services operating in sequence: a preprocessor, which consumes raw EVE JSON records from the `on-premise-info` Kafka topic, extracts the 34-dimensional feature vector from each flow record, and forwards the result to the `preprocessed-data` topic; a detector, which consumes those feature vectors, runs inference against the loaded model, and publishes scored records to the `alerts` topic; and a dashboard, which consumes the alert stream and renders a live monitoring interface. The model artifact and its associated threshold file are bundled into the detector image at build time. Deployment onto the CYberSynchrony Kubernetes infrastructure is managed by eBOS, which operates both the cluster and the central Kafka broker through which all inter-component communication flows.

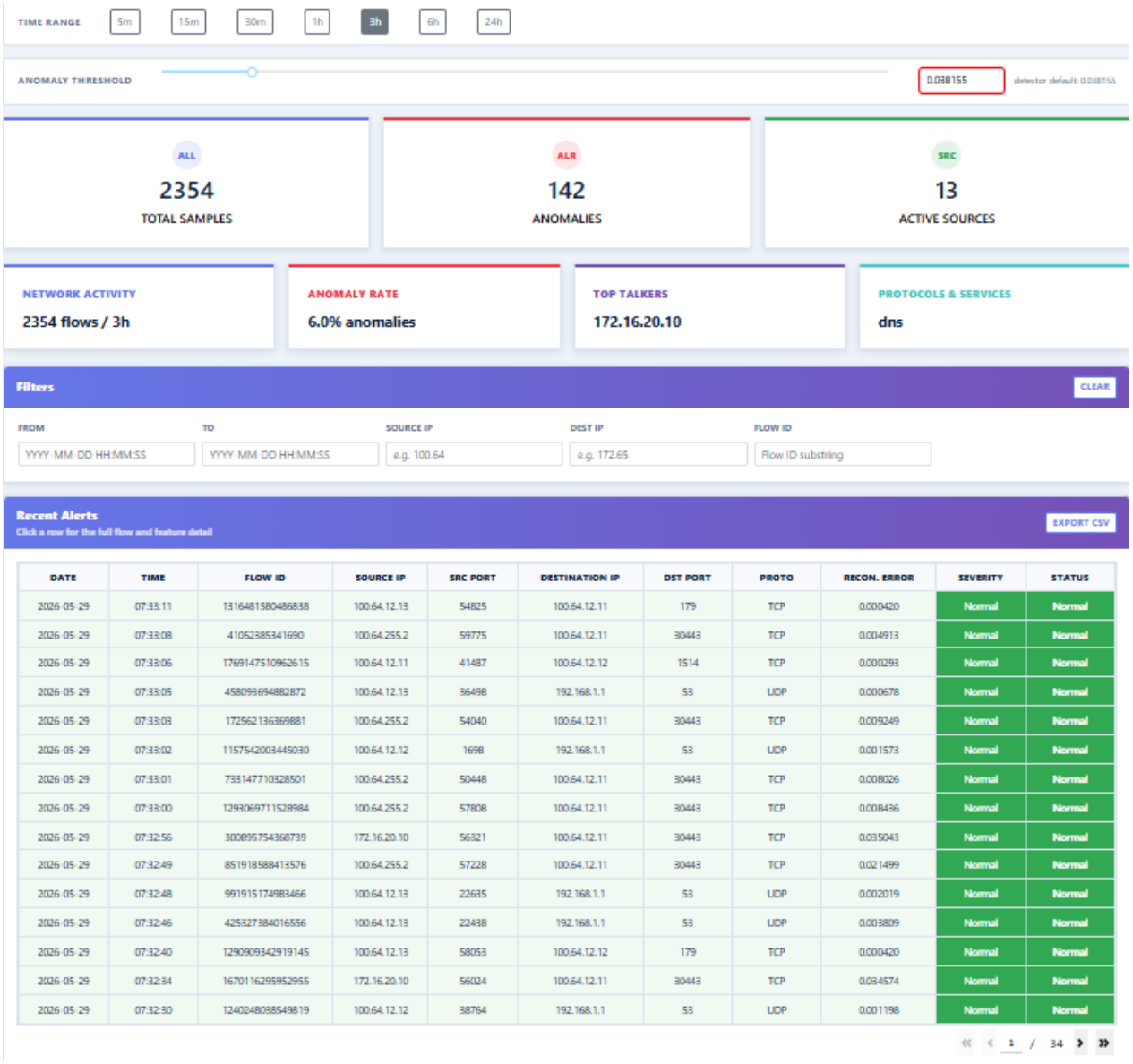
5.2.4 Monitoring dashboard

The dashboard provides a real-time view of the Anomaly Detector's output, consuming the `alerts` topic and refreshing continuously as new scored records arrive. It presents aggregate statistics — total flows processed, anomaly count, anomaly rate, and active source count — alongside an alert table showing recent flows colour-coded by severity. Severity is derived from the ratio of a flow's reconstruction error to the configured threshold and ranges across five levels: Normal, Low, Medium, High, and Critical. Analysts can drill into any individual record to inspect the full flow metadata, and the table can be exported for offline analysis. The dashboard is the

© CYberSynchorny, 2026

primary operational interface for security personnel interacting with the Anomaly Detector during pilot execution.

Figure 13 illustrates the main dashboard view with the alert table and aggregate metrics; Analytics panels available for deeper inspection are also available, including network activity over time (Figure 14), anomaly rate trends (Figure 15), top talker breakdowns (Figure 16) and flows w.r.t. protocols & services (Figure 17).



DATE	TIME	FLOW ID	SOURCE IP	SRC PORT	DESTINATION IP	DST PORT	PROTO	RECON. ERROR	SEVERITY	STATUS
2026-05-29	07:33:11	1316481580486838	100.64.12.13	54825	100.64.12.11	179	TCP	0.000420	Normal	Normal
2026-05-29	07:33:08	41052385341690	100.64.255.2	59775	100.64.12.11	30443	TCP	0.004913	Normal	Normal
2026-05-29	07:33:06	1769147510962615	100.64.12.11	41487	100.64.12.12	1514	TCP	0.000293	Normal	Normal
2026-05-29	07:33:05	4580936948882872	100.64.12.13	36498	192.168.1.1	53	UDP	0.000678	Normal	Normal
2026-05-29	07:33:03	1725621366369881	100.64.255.2	54040	100.64.12.11	30443	TCP	0.009249	Normal	Normal
2026-05-29	07:33:02	1157542003445030	100.64.12.12	1698	192.168.1.1	53	UDP	0.001573	Normal	Normal
2026-05-29	07:33:01	7331477103285O1	100.64.255.2	50448	100.64.12.11	30443	TCP	0.008026	Normal	Normal
2026-05-29	07:33:00	1293069711528984	100.64.255.2	57808	100.64.12.11	30443	TCP	0.008436	Normal	Normal
2026-05-29	07:32:56	300895754368739	172.16.20.10	56321	100.64.12.11	30443	TCP	0.035043	Normal	Normal
2026-05-29	07:32:49	851918588413576	100.64.255.2	57228	100.64.12.11	30443	TCP	0.021499	Normal	Normal
2026-05-29	07:32:48	991915174983466	100.64.12.13	22635	192.168.1.1	53	UDP	0.002019	Normal	Normal
2026-05-29	07:32:46	425327384016556	100.64.12.13	22438	192.168.1.1	53	UDP	0.003809	Normal	Normal
2026-05-29	07:32:40	1290909342919145	100.64.12.13	58053	100.64.12.12	179	TCP	0.000420	Normal	Normal
2026-05-29	07:32:34	1670116295952955	172.16.20.10	56024	100.64.12.11	30443	TCP	0.034574	Normal	Normal
2026-05-29	07:32:30	1240248038549819	100.64.12.12	38764	192.168.1.1	53	UDP	0.001198	Normal	Normal

Figure 13. Detector's main dashboard

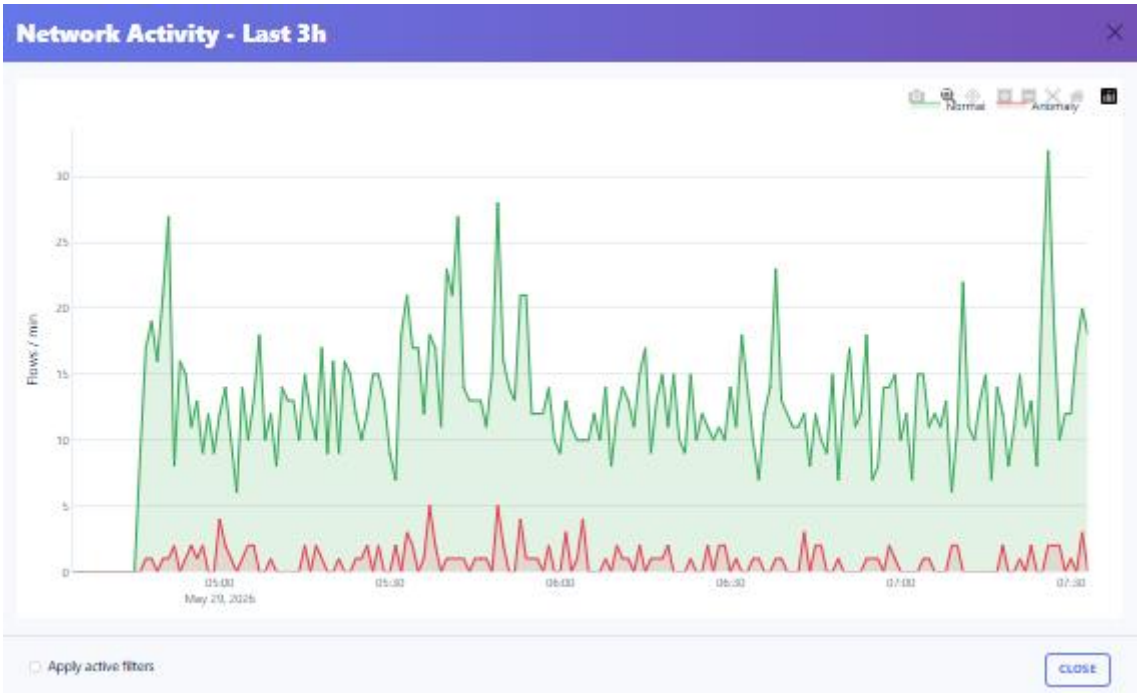


Figure 14. Network activity

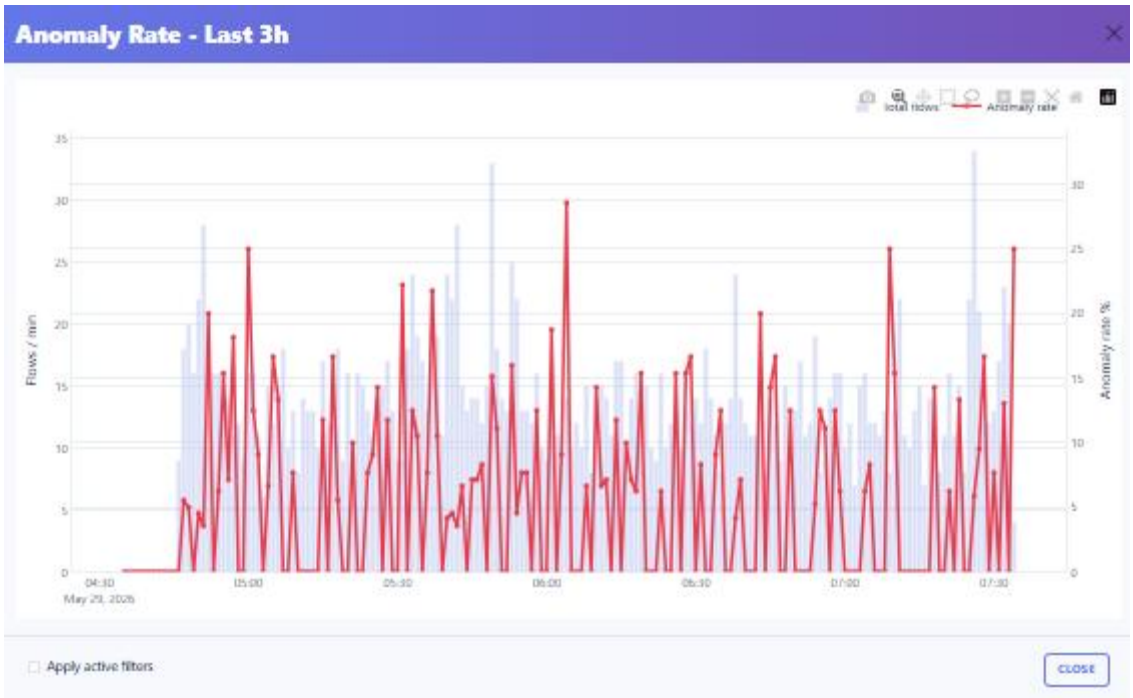


Figure 15. Anomaly rate

© CYberSynchorny, 2026

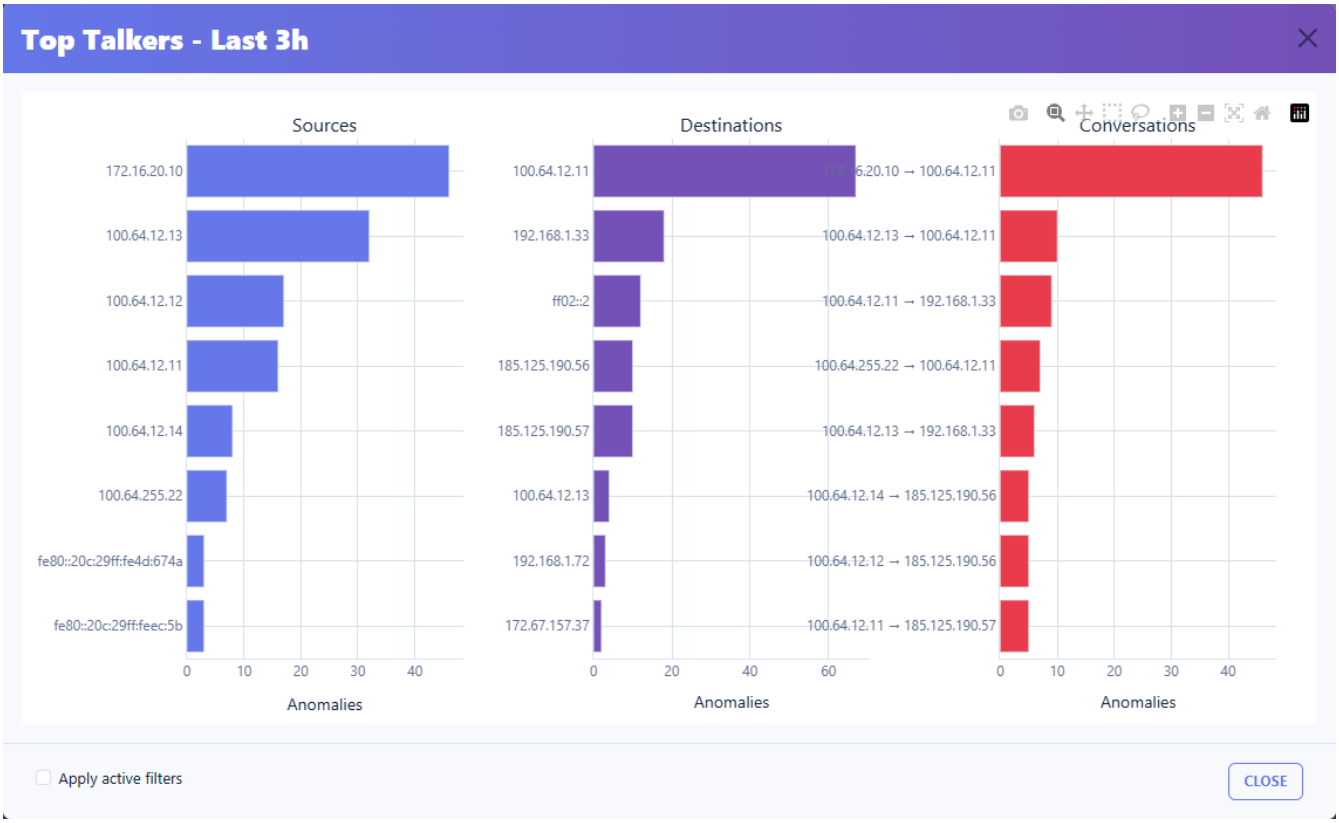


Figure 16. Top talker IPs

Figure 17. Flows per protocol

© CYberSynchorny, 2026

6 WEKA

6.1 Updates since V1

WEKA has not been modified since D4.1. WEKA is a standalone tool, and the description, architecture, and technology stack documented there remain accurate. Partners can use it to analyse their data for potential anomalies. We have created a detailed manual for any partner, internal or external, to use the tool. A small version of the manual is presented in the next subsection.

6.2 Integration, interoperability and deployment

A simple manual on using WEKA in step-by-step mode is presented here. This only covers the explorer mode of WEKA, although there are more modes that it can be used depending on the user's experience (Figure 18).

Click the "***Explorer***" button to launch the Weka Explorer. This GUI lets you load datasets and run classification algorithms. It also provides other features like data filtering, clustering, association rule extraction and visualization.

Figure 18. WEKA GUI

WEKA's main graphical user interface, the Explorer, gives access to all its facilities using menu selection and form filling (Figure 19). It is illustrated in the following Figure. To begin, there are six different panels, selected by the tabs at the top, corresponding to the various data mining tasks supported by WEKA. Further panels can be made available by installing the appropriate packages.

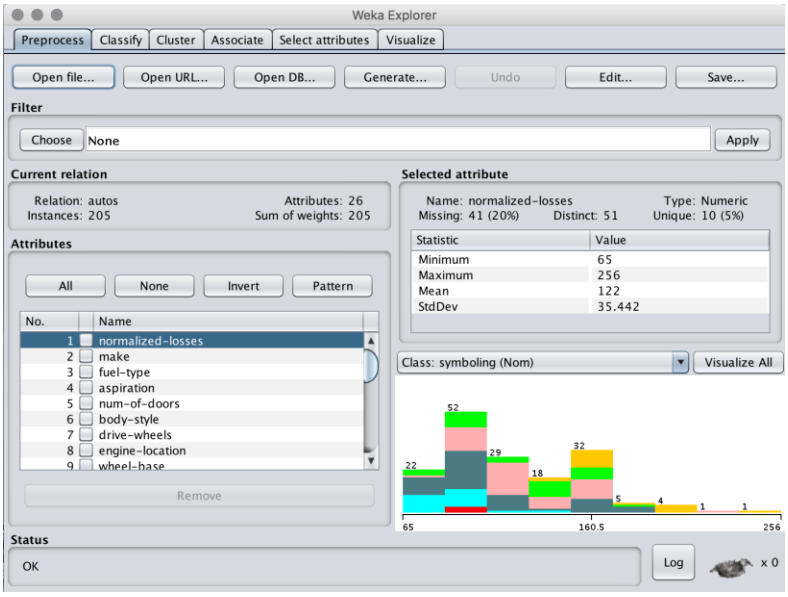


Figure 19. Explorer interface

© CYberSynchorny, 2026

Suppose you have some data and you want to build a decision tree from it. First, prepare the data, then fire up the Explorer and load it. Next, you select a decision tree construction method, build a tree, and interpret the output. It is easy to do it again with a different tree construction algorithm or a different evaluation method. In the Explorer you can flip back and forth between the results you have obtained, evaluate the models that have been built on different datasets, and visualize graphically both the models and the datasets themselves—including any classification errors the models make (Figure 20).

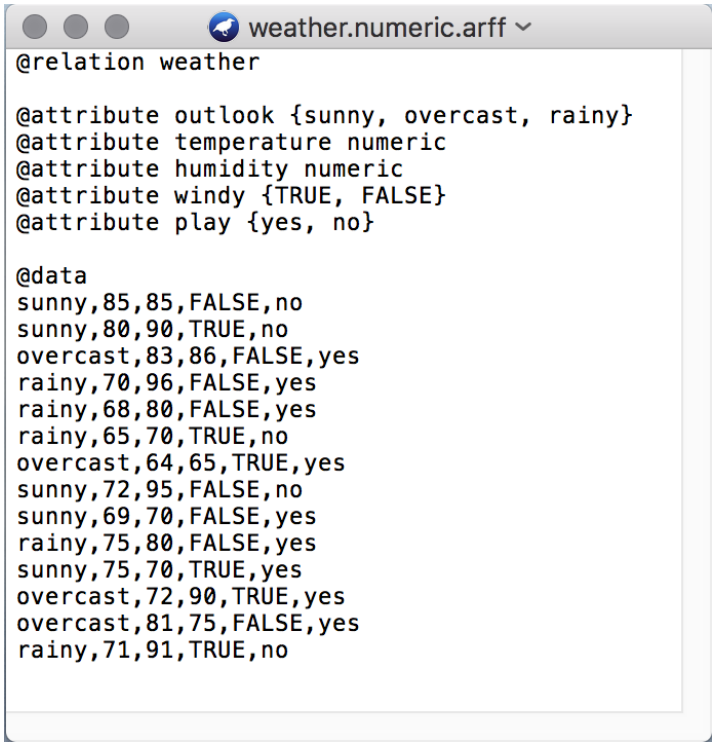


Figure 20. Sample data about weather

The data is often presented in a spreadsheet or database. However, WEKA's native data storage format is ARFF. You can easily convert from a spreadsheet to ARFF. The bulk of an ARFF file consists of a list of the instances, and the attribute values for each instance are separated by commas. Most spreadsheet and database programs allow you to export data into a file in comma-separated value (CSV) format as a list of records with commas between items. Having done this, you need only load the file into a text editor or word processor; add the dataset's name using the @relation tag, the attribute information using @attribute, and a @data line; and save the file as raw text.

At the top of the classification section is the Classifier box. This box contains a text field that displays the name of the currently selected classifier, and its options. Clicking on the text box with the left mouse button brings up a GenericObjectEditor dialog box, just the same as for filters, that you can use to configure the options of the current classifier. With a right click (or Alt+Shift+left click) you can once again copy the setup string to the clipboard or display the properties in a GenericObjectEditor dialogue box. The Choose button lets you select one of the classifiers available in WEKA.

© CYberSynchorny, 2026

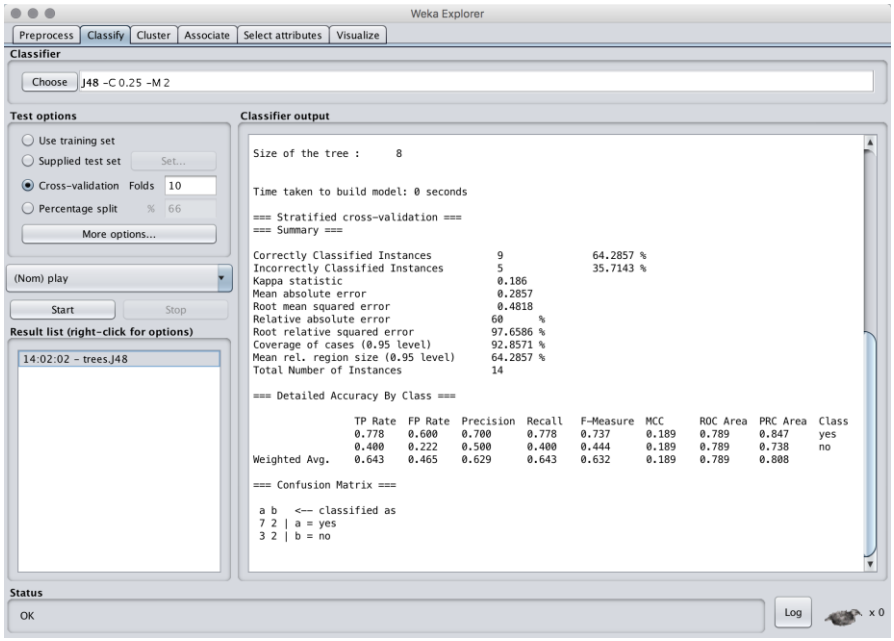


Figure 21. The Classify tab

WEKA's visualization section allows you to visualize 2D plots of the current relation.

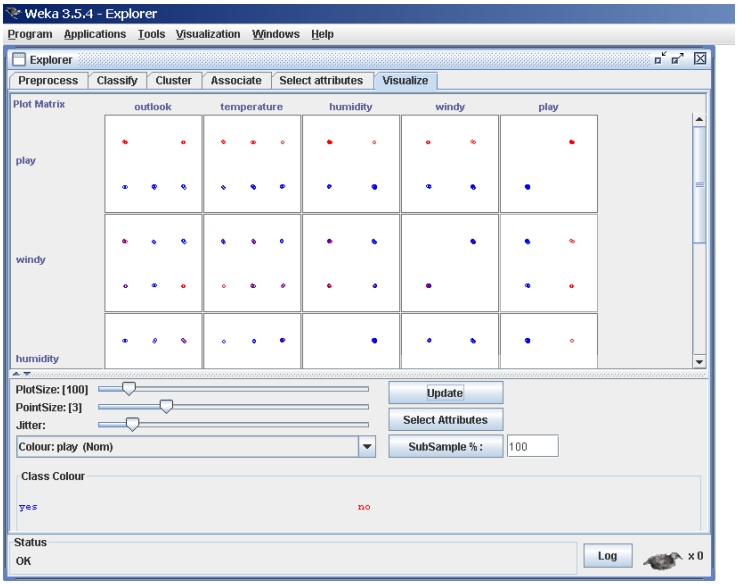


Figure 22. The Visualize tab

When you select the Visualize panel, it shows a scatter plot matrix for all the attributes, colour coded according to the currently selected class. It is possible to change the size of each individual 2D plot and the point size, and to randomly jitter the data (to uncover obscured points). It is also possible to change the attribute used to colour the plots, to select only a subset of attributes for inclusion in the scatter plot matrix, and to sub sample the data.

© CYberSynchorny, 2026

7 trustSENSE

7.1 Updates since V1

We have decided to substitute HRM with trustSENSE. trustSENSE is an assessment tool that helps organizations evaluate the maturity and readiness of AI and cybersecurity teams. It provides actionable insights to improve trustworthiness, manage risks, and enhance the resilience and security of AI ecosystems across technical, user, and defence roles. There are several user group roles to pick: SOC analyst / IT administrator / policymaker / trainee / pilot user, etc. This substitution is justified because trustSENSE offers a broader and more directly applicable framework for assessing both technical capabilities and organizational readiness in AI-enabled cybersecurity contexts. Its role-based structure also makes it more suitable for capturing the diverse perspectives and responsibilities of the different stakeholders involved.

7.2 Integration, interoperability and deployment

The trustsense tool is fully integrated into the Cybersychrony platform through a dedicated link. After visiting the link, the Risk assessor must select one of the provided team roles.

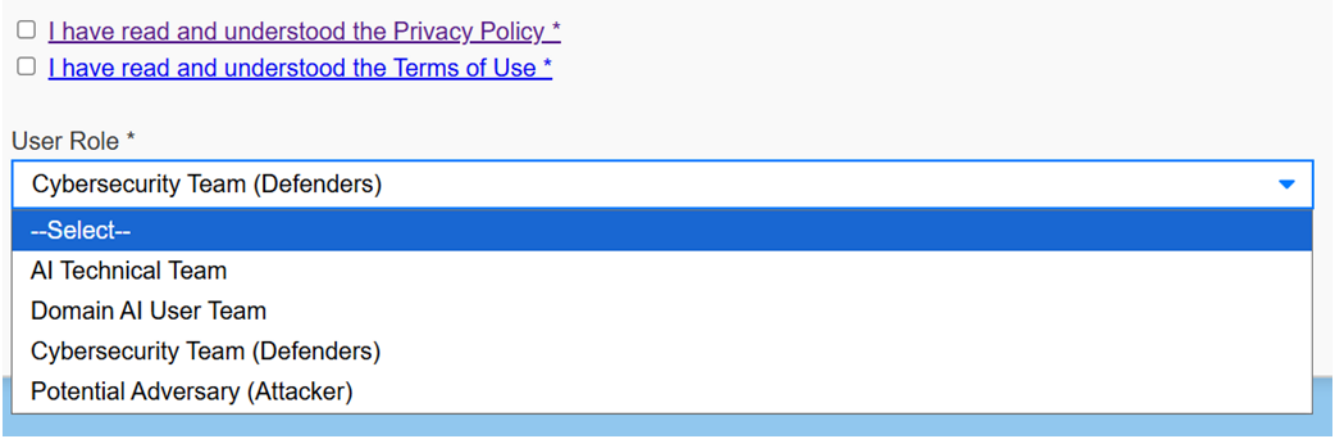


Figure 23. Initial screen of trustSENSE

After the user selects the role that best matches their position in the company, the assessment page is shown. Here, the user must pick the best option for each of the 23 statements shown after consensus with the team. For every such statement, 5 options are given, ranging from strongly agree to strongly disagree.

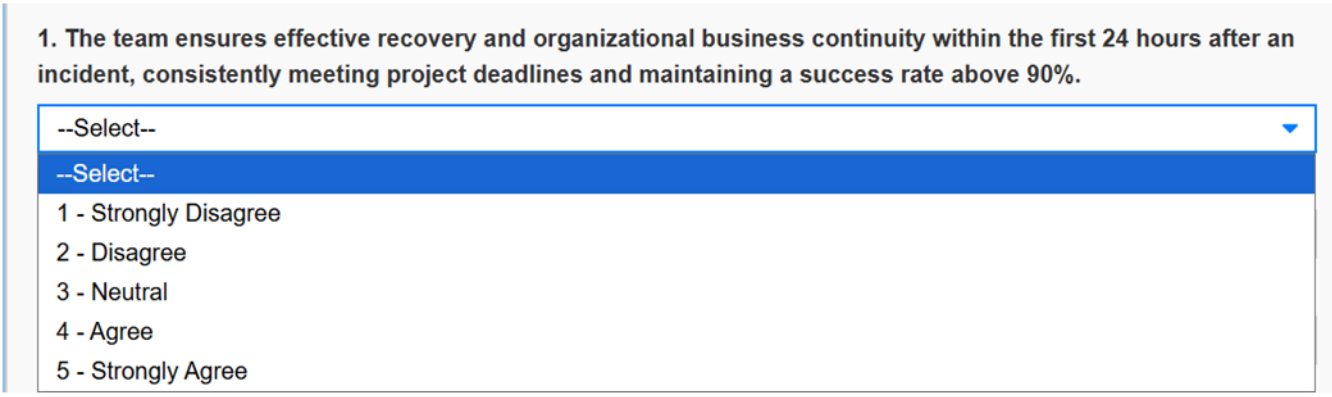


Figure 24. Assessment page with the 23 questions

Once all questions are answered, the user presses submit and the tool calculates the team's Trustworthiness Level Result (an example is shown in Figure 25).

© CYberSynchorny, 2026

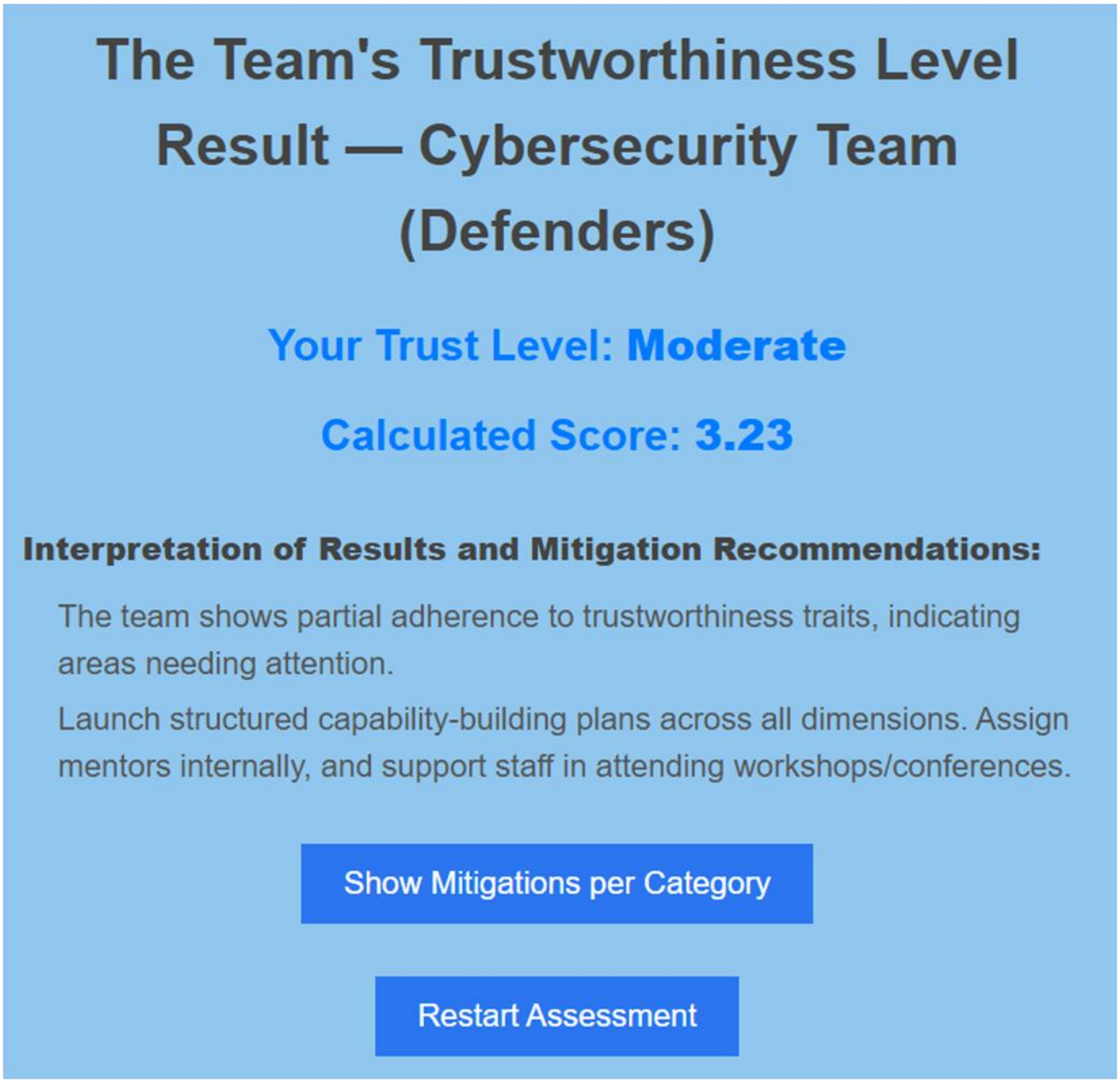


Figure 25. Team's Trustworthiness Level Result

This assessment provides a total score out of 5 (maximum score), a description of the team's readiness and basic advice. To view detailed proposed mitigation actions per category, the user must click the "Show Mitigations per Category" button. Once the button is clicked, mitigation actions for 12 categories are shown to the user (example mitigations are shown in Figure 26).

© CYberSynchorny, 2026

Figure 26. Example Mitigation actions proposed by the trustSENSE tool

Important information:

- A risk assessor gathers their team to answer each question through consensus, as the tool is designed for a collaborative team approach rather than a single user.
- The tool doesn't store any information about the user or the outcome for privacy preservation and GDPR compliance.

The tool can be reused as many times as needed in order to assess other roles in the company or to record the progress of the company after some mitigation actions have already been taken. Since it is now fully integrated into the platform, it is used by partners who lead the use cases to assess the level of readiness of the IT teams in order to help shape a better training program for each organization. TrustSENSE will be used along with dedicated questionnaires that the CYberSynchrony consortium has created for the users of the systems of the three use cases. Moreover, trustSENSE will be offered to all external parties that participate in WP7 through the open calls.

© CYberSynchorny, 2026

8 KPIs Mapping

This chapter maps monitoring suite components documented in this deliverable to the project KPIs to which they contribute. Measured KPI values are reported through WP8 deliverables. Pilot operations start from M25, with the first inputs from Wazuh, Suricata, and the Anomaly Detector expected during M25 and M26.

8.1 KPI1.1: Number of Threat Assessments Carried Out

KPI ID	**KPI Name**	**Owner**
1.1	Number of Threat Assessments Carried Out	IANUS
Target		
More than 36 threat assessments across the project. The row schema, evidence requirements, and reporting path (D8.2) are defined in the project KPI register.		
Measurement Methodology		
Components involved	**Participating partners & Role in KPI Measurement**	**Evidence collection process & validation approach**
MISP	IANUS uses MISP to identify, enrich and validate threat intelligence indicators, such as malicious IPs, domains, URLs, hashes, malware-related indicators and known threat events relevant to pilot assets.	MISP event exports, IoC lookup results, screenshots, API response extracts, matched IoC records, event IDs and analyst notes explaining why the IoC is relevant to the pilot/use case.
Wazuh	IANUS uses Wazuh to collect and analyse endpoint, system, security configuration, vulnerability, authentication, rootcheck and alert data from the pilot environments.	Wazuh alert IDs, dashboard screenshots, JSON alert exports, rule IDs, severity levels, affected agent/asset, timestamp and analyst validation
Suricata	IANUS uses Suricata to detect suspicious network traffic, IDS rule matches, command-and-control patterns, abnormal connections and network-level threat indicators in pilot environments.	Suricata alert records, Wazuh-normalised Suricata alerts, eve.json extracts, source/destination IPs, ports, protocols, rule signatures, severity and dashboard screenshots.
Anomaly Detector IDS	MAG uses the Anomaly Detector IDS to identify unusual traffic windows or behavioural deviations from the learned network baseline, based on Suricata flow features and shared data pipelines.	Anomaly alert payloads, anomaly score, reconstruction error, affected flow window, contributing features such as protocol, ports, packet counts, byte counts, flow duration and peer pairs, plus any already-correlated Wazuh, Suricata or MISP evidence
Kitsune Risk/Monitoring	MER uses Kitsune Search / Monitoring to identify threat intelligence, exposed information, suspicious external references, leak-related indicators or	Kitsune reports, screenshots, monitoring results, search outputs, linked external

© CYberSynchorny, 2026

	abnormal risk signals relevant to pilots and use cases.	references, affected organisation/use case, analyst notes and recommended actions.

Methodology Description

The responsible owner will create a Threat Assessment Register, where each valid row counts as one threat assessment when a concrete threat identified by the deployed tools is analysed in relation to assets, systems, networks, services, users, or data belonging to a specific pilot or use case. A raw alert alone does not automatically count as a threat assessment. It counts only when the responsible partner reviews the finding, validates its relevance, documents the affected pilot asset, assigns severity and likely impact, links supporting evidence, and records recommended mitigation or response actions. Minimum evidence for each threat assessment will include date, pilot/use case, partner, tool used, threat type, affected asset or service, severity, likelihood/impact, recommended action, validation status, and evidence link.

The current platform components produce alerts, indicators, anomalies or intelligence findings. The responsible partner checks whether the finding is relevant to the pilot, what asset or service it affects, how serious it is, and what should be done. Each validated and documented case counts as one threat assessment. Threat assessments may come from individual tools or from correlation between multiple tools. A higher-confidence threat assessment may also result from combined evidence, for example when a Suricata alert matches a MISP IoC and a concurrent Wazuh event is observed on the same pilot asset. For example, the Anomaly Detector contributes one threat assessment row per validated anomaly, as it consumes Suricata flow features through the shared data pipeline and flags network traffic windows that deviate from the learned baseline. The responsible analyst reviews the flagged anomaly, checks the contributing flow features such as protocol, ports, byte counts, packet counts, flow duration and communicating peers, and correlates the finding with Suricata signatures, Wazuh alerts, MISP IoCs or other available evidence. If the anomaly is considered a real or relevant threat, it is recorded in the Threat Assessment Register.

KPI Progress Status

Current Value/Target	**Date of Measurement**	**Status**	**Next Actions** (if required)
0/36	31/05/2026	On Track	Create the Threat Assessment Register, begin recording validated findings once pilot data collection starts.

Challenges, Deviations and Corrective Actions

A key challenge for this KPI is to avoid counting raw tool alerts as completed threat assessments. To address this, only findings that have been reviewed, validated, linked to a specific pilot/use case, assigned severity and impact, supported with evidence, and accompanied by a recommended action will be counted. Where several tools report the same event, the outputs will be correlated and recorded as one higher-confidence threat assessment when they refer to the same threat, asset and time window.

© CYberSynchorny, 2026

9 Conclusions

At M25, the headline result for D4.3 is integration of the five monitoring suite components, which no longer operate as separate tools but share two common layers. Wazuh produces a single alert stream that merges Suricata IDS alerts, Wazuh-native endpoint events, and MISP threat-intelligence enrichment, giving analysts one place to review network, host, and threat-context information together. In parallel, a Kafka-based event bus distributes Suricata flow telemetry to the Anomaly Detector, which runs as a three-stage containerised pipeline (preprocessing, inference, and dashboard) on the project's Kubernetes infrastructure, with a live view of anomalous traffic.

WEKA and trustSENSE sit outside these shared layers by design. WEKA remains unchanged from v1 and is still available to any partner as a standalone forensic analysis tool, now with a published user manual. trustSENSE has replaced HRM as the project's participatory team-readiness assessment: it runs independently of the platform's data pipeline and use case leads will be using it to gauge IT team readiness ahead of the training programmes planned for the pilots.

A few items remain to be addressed aheMad of pilot operations. On the Wazuh side, infrastructure hardening and Keycloak-based SSO configuration are planned. The interface between the Wazuh alert stream and the blockchain integrity component is specified in Chapter 3 (Table 5), with validation against an actual deployment to follow. For the Anomaly Detector, performance evaluation against simulated attack scenarios is scheduled for the pilot phase.

Pilot operations begin from M25. Chapter 8 opens the KPI mapping for the monitoring suite with KPI1.1 (Number of Threat Assessments Carried Out) fully specified; the remaining relevant KPIs will be mapped as the pilots generate the first evidence. Measured KPI values and pilot execution results will be reported through WP8 (D8.2, D8.4).

© CYberSynchorny, 2026