Harmonizing People, Processes, and Technology for Robust Cybersecurity

D3.3 Enhanced Threat Intelligence System 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	D3.3 Enhanced Threat Intelligence System v2		
Work Package	WP3 Comprehensive Threat Intelligence and Customized Risk Evaluation (CYBERRA) and Cybersecurity Resilience and Incident Response Enhancement (CYRESCUE)		
Contractual due date	31/05/2026	**Actual submission date**	30/06/2026
Type	Other	**Dissemination Level**	Public
Lead Beneficiary	BYTE		
Responsible Author	George Karavokiros (BYTE)		
Contributions from	Andrea Purificato (MER), Evangelos Michos (IANUS), Vasilis Keleris (MAG)		

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	20/02/2026	ToC, initial structure	George Karavokiros (BYTE), Vasilis Keleris (MAG)
0.2	23/03/2026	Integration of ToC and scoping	George Karavokiros (BYTE)
0.3	21/04/2026	Technical contributions – MISP	Evangelos Michos (IANUS)
0.4	18/05/2026	Technical contributions – Kitsune	Andrea Purificato (MER)
0.5	22/05/2026	Technical contributions – UFT	George Karavokiros (BYTE), Vasilis Keleris (MAG)
0.6	15/06/2026	Additional Contributions Peer - Review 1 & 2	George Karavokiros (BYTE), Vasilis Keleris (MAG), Evangelos Michos (IANUS), Andrea Purificato (MER)
0.7	19/06/2026	Addressing review comments	George Karavokiros (BYTE)
0.8	29/06/2026	Final draft addressing comments after the internal peer review	Kostas Andrikos (MAG) Leandros Maglaras (TRUST) George Karavokiros (BYTE)
1.0	30/06/2026	QA and Final version ready for submission	Eleni Keliri (eBOS), Violeta Vasileva (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.

[1] According to CYberSynchrony's Quality Assurance Process for reports:

- 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

© CYberSynchrony, 2024

Table of Contents

Executive Summary 8
1 Introduction 9
1.1 Mapping CYberSynchrony Outputs 9
1.2 Linkage to other project outputs 10
1.3 Deliverable Structure 10
2 Position in CYberSynchrony Architecture 12
3 MISP 13
3.1 Updates since v1 13
3.2 Integration, interoperability and deployment 13
3.2.1 Connectivity Test 13
3.2.2 Feed Database 14
3.2.3 Integration efforts 14
3.2.4 Added value of integration activities 16
3.2.5 Dashboards and Analytics 18
4 Kitsune Search 24
4.1 Updates since v1 24
4.2 Integration, interoperability and deployment 24
Graphical User Interface 24
5 Unalterable Forensic Techniques adaptation (UFT) 26
5.1 Updates since v1 26
5.2 Design rationale and threat model 27
5.2.1 Goal and threat model 27
5.2.2 Why a simple file fingerprint is insufficient 27
5.2.3 Unification point: where Wazuh and Suricata streams converge 27
5.3 Cryptographic design 28
5.3.1 Cryptographic primitive: HMAC-SHA256 28
5.3.2 Per-agent secret_key lifecycle 28
5.4 End-to-end alert pipeline 29
5.4.1 Agent-side signing flow (cybersync4) 29
5.4.2 Manager-side verification flow (cybersync1) 29
5.4.3 Tamper-detection rules and SOC visibility 29
5.4.4 Anchoring verified alerts into Hyperledger Fabric 30
5.5 Implementation, security and deployment 30
5.5.1 Implementation artefacts 30
5.5.2 Security analysis and residual risks 31
5.5.3 Interoperability and deployment status 31
6 KPIs Mapping 32
6.1 KPI1.3: Threat Intelligence Platform Effectiveness 32
6.2 KPI1.5: Threat Intelligence Platform Security 34
7 Conclusions 37
Annex I: PoC and Prototypes (Kitsune Search) 38
Integrated Kitsune Front-End and Intelligence Reach 38

© CYberSynchrony, 2024

List of Figures

Figure 1. Updated positioning of D3.3 components in CYberSynchrony architecture. 12
Figure 2. MISP feeds enabled 14
Figure 3. MISP Intelligence Flow 15
Figure 4. MISP Dashboard - Events 19
Figure 5. MISP Dashboard – View Organization 19
Figure 6. MISP Dashboard - Investigate incident 20
Figure 7. MISP Dashboard – Event attributes 20
Figure 8: MISP Dashboard - Event Graph 20
Figure 9: MISP Dashboard - Adding Tags 21
Figure 10: MISP Dashboard - Add Galaxy Cluster 21
Figure 11: Populate from - Dropdown Menu 22
Figure 12: Populate from - Freetext Import Tool 22
Figure 13. Wazuh dashboard with MISP metadata 23
Figure 14: Kitsune Login 38
Figure 15: Kitsune Homepage 39
Figure 16: Kitsune Underground Search 40

List of Tables

Table 1: Adherence to CYberSynchrony's GA 9
Table 2: Links to other Project Outputs/Work 10
Table 3: Wazuh alerts used for MISP lookup 15
Table 4: Before and After MISP enrichment 16
Table 5: MISP IOC Categories 18
Table 6: Wazuh – MISP API communication 22

© CYberSynchrony, 2024

Acronyms and abbreviations

Abbreviation / Term	Description
AI	Artificial Intelligence
API	Application Programming Interface
ASA	Attack Surface Assessment
CIRCL	Computer Incident Response Center Luxembourg
CISA	Cybersecurity and Infrastructure Security Agency
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
CWE	Common Weakness Enumeration
DCO	Defensive Cyber Operations
DNS	Domain Name System
DPM	Data Preparation Module
FIM	File Integrity Monitoring
FIPS	Federal Information Processing Standards
GA	Grant Agreement
GUI	Graphical User Interface
HMAC	Hash-based Message Authentication Code
HTTP	Hypertext Transfer Protocol
IDS	Intrusion Detection System
IPC	Inter-Process Communication
JSON	JavaScript Object Notation
KEV	Known Exploited Vulnerabilities
KPI	Key Performance Indicator
MAC	Message Authentication Code
MISP	Malware Information Sharing Platform
MITRE	MITRE Corporation
NVD	National Vulnerability Database
OSINT	Open Source Intelligence
REST	Representational State Transfer
RFC	Request for Comments

© CYberSynchrony, 2024

SCA	Security Configuration Assessment
SHA	Secure Hash Algorithm
SIEM	Security Information and Event Management
SOC	Security Operations Center
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UFT	Unalterable Forensic Techniques
UI	User interface
UUID	Universally Unique Identifier
VM	Virtual Machine
WP	Work Package

© CYberSynchrony, 2024

Executive Summary

D3.3 Enhanced Threat Intelligence System v2 is the second iteration of deliverable D3.1. It documents integration and interoperability progress for three WP3 components since D3.1: MISP, Kitsune Search, and the Unalterable Forensic Techniques adaptation (UFT).

MISP is now connected to the Wazuh/Suricata monitoring pipeline. Indicators extracted from Wazuh-enriched Suricata alerts (source and destination IPs, hostnames, URLs, and file hashes) are queried against a local database of roughly 170,000 IoCs from the CIRCL OSINT feed. Detection events are no longer handled in isolation; each one is correlated against that intelligence base, and SOC analysts see the matched IoC type, threat event, source organisation, and threat level on the Wazuh Dashboard alongside the alert.

Kitsune Search has moved from a standalone backend API to a web component inside the CYberSynchrony centralised dashboard. Authorised analysts now reach Kitsune's dark web and underground intelligence capabilities (threat actor tracking, botnet exposure, data leak retrieval) through the dashboard UI, with no need to interact with the raw API.

The Unalterable Forensic Techniques component was rebuilt. V1 computed a single SHA-256 hash per batch of logs and anchored it in Hyperledger Fabric. Three problems eventually made this unworkable: per-alert tampering was undetectable at batch granularity; there was no producer authentication; and the scheme assumed identical byte sequences on agent and cloud, which Wazuh's data plane does not guarantee. Version 2 switches to per-alert HMAC-SHA256. A signer sidecar on the on-premises node attaches a keyed MAC to each alert before it enters the Wazuh transport. On the cloud side, a verifier checks the MAC before anchoring the alert to the Fabric ledger, alerts that fail are dropped, and any mismatch raises a high-severity tamper detection rule in Wazuh.

© CYberSynchrony, 2024

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, in 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			
D3.3	Enhanced Threat Intelligence System v2	Chapters 1-6, Annex I	This deliverable presents the second-version integration and interoperability state of the WP3 threat intelligence components. It documents the Wazuh/Suricata–MISP enrichment workflow, the Kitsune Search front-end integration, and the redesigned per-alert HMAC-SHA256 forensic integrity pipeline with Hyperledger Fabric anchoring.
TASKS			
T3.1	T3.1 – Cyber-Threat Intelligence Platforms Deployment and Fine-Tuning	Chapter 3	Describes the deployment and fine-tuning of MISP as a structured and sharable threat intelligence platform. It highlights MISP's integration with other tools to ingest and disseminate cyber threat information efficiently
T3.2	T3.2 – Adopting Unalterable Forensic Techniques	Chapter 5,	Details the architecture and design of UFT processes, which ensure forensic soundness by protecting the integrity of cyber evidence before it enters the analysis pipeline
T3.3	T3.3 – Crafting Personalized Risk Assessment Tools	Chapter 4, Annex I	Presents Kitsune Search, a system enabling asset-aware threat intelligence that tailors risk correlation to specific digital assets, enhancing the precision of threat assessments

© CYberSynchrony, 2024

1.2 Linkage to other project outputs

D3.3 builds directly on D3.1, which established the baseline architecture and standalone deployment of MISP, Kitsune Search, and the UFT Data Preparation Module. The technical descriptions in D3.1 remain the reference for component-level specifications; D3.3 does not repeat them but documents what has changed and how each component communicates with the rest of the platform.

Table 2: Links to other Project Outputs/Work

WP Number	Task Number	Deliverable Number related	Content
WP2	T2.3, T2.4	D2.1	The architectural principles and system requirements defined in T2.4 guided the integration approach for MISP, UFT, and Kitsune Search. These components were designed to align with the modular, interoperable architecture established in D2.1.
WP3	T3.1, T3.2, T3.3	D3.1	D3.1 provides the foundational architecture and component specifications on which all v2 integration work in D3.3 builds. Changes documented in D3.3 are explicitly scoped against D3.1 baselines.
WP4	T4.1	D4.1, D4.3	The Wazuh/Suricata monitoring pipeline deployed and documented in D4.1 and D4.3 is the upstream source of alerts consumed by both the MISP enrichment workflow and the UFT signing chain. The enriched MISP output is also available to the AI Anomaly Detector pipeline documented in D4.3.
WP5	T5.1	D5.1, D5.3	The distributed ledger technologies developed under T5.1 are fundamental to the UFT solution. They enable secure hashing and traceability of security log data, ensuring immutable records for forensic use by IDS components.
WP8	T8.1	D8.1	The design and development of threat intelligence and unalterable forensic components in D3.1 were guided by the need for seamless integration into the CYberSynchrony platform. The work performed under WP3 took into consideration the technical and operational requirements of the overall integration effort, ensuring that the developed tools align with the platform's use case-driven validation approach defined in WP8.

1.3 Deliverable Structure

This document is organised as follows.

- **Chapter 2** positions the D3.3 components within the broader CYberSynchrony platform architecture.
- **Chapter 3** covers MISP (IANUS). It documents the updates since v1, including the Wazuh/Suricata integration layer, the CIRCL OSINT feed configuration, the IoC lookup and enrichment logic, and the resulting SOC analyst view on the Wazuh Dashboard.
- **Chapter 4** covers Kitsune Search (MER). It documents the transition from a standalone backend service to an API-driven component with a web-based front-end integrated into the CYberSynchrony dashboard.

© CYberSynchrony, 2024

- **Chapter 5** covers the Unalterable Forensic Techniques adaptation (BYTE/MAG). It documents the redesign from batch-level SHA-256 hashing to a per-alert HMAC-SHA256 scheme, covering design rationale, cryptographic design, the end-to-end signing and verification pipeline, and Hyperledger Fabric anchoring.
- **Chapter 6** maps how the documented components contribute to project's KPI's, and outlines the measurement methodology and evidence sources.
- **Chapter 7** presents the conclusions

© CYberSynchrony, 2024

2 Position in CYberSynchrony Architecture

Figure 1 presents the position of D3.3's components to the project's updated architecture. MISP's connection to the Wazuh/Suricata monitoring flow has gone from planned to validated and operational since v1 (Chapter 3), and Kitsune Search now reaches users through a dashboard-integrated front-end instead of a backend-only API (Chapter 4). The UFT pipeline replaced batch-level SHA-256 hashing with per-alert HMAC-SHA256 signing and Hyperledger Fabric anchoring (Chapter 5). The Wazuh Manager, Wazuh Indexer, and Kafka broker are documented in D4.1 and D4.3. They appear here for context only.

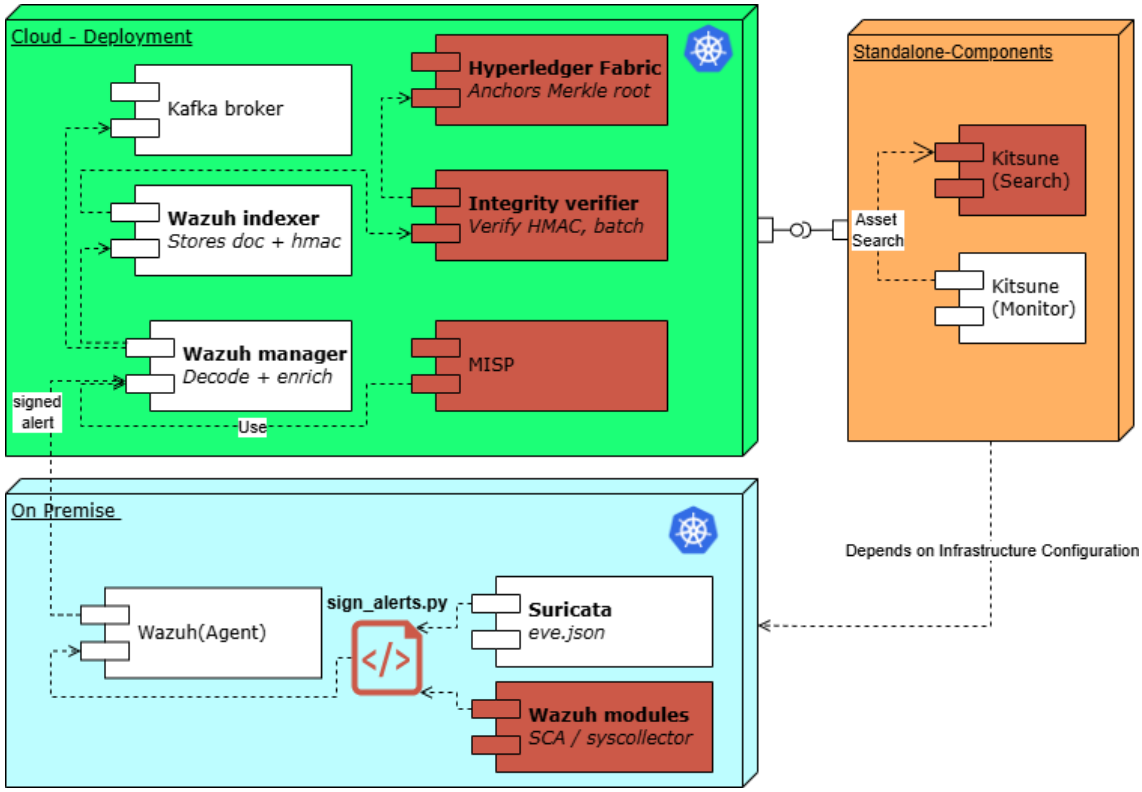

Figure 1. Updated positioning of D3.3 components in CYberSynchrony architecture.

© CYberSynchrony, 2024

3 MISP

3.1 Updates since v1

The core MISP technologies, architecture and functional description remain as described in **D3.1 Enhanced Threat Intelligence System v1**, where MISP was described as the CYberSynchrony threat intelligence platform responsible for aggregating, analysing, correlating and sharing structured cyber threat intelligence, including IoCs, events, attributes, galaxies and taxonomies. D3.1 defines MISP as a cloud-based threat intelligence hub that exposes enriched IoCs and contextual metadata through REST APIs and supports integration with tools such as Wazuh and Suricata.

The main update since v1 concerns the integration layer. **MISP has now been connected with the Wazuh/Suricata monitoring flow**, allowing alerts and network security events generated by Wazuh and Suricata to be correlated against MISP IoCs and external threat intelligence feeds. Through this workflow, indicators such as suspicious IP addresses, domains, URLs and hashes extracted from Wazuh/Suricata alerts are checked against MISP, enabling the platform to enrich local security alerts with external threat context, such as known malicious infrastructure, malware associations, threat events, source organisations and threat levels. This integration enhances the value of the Wazuh and Suricata data by transforming local detection outputs into enriched threat intelligence records. This is also visible through the Wazuh Dashboard, enabling SOC analysts to investigate alerts with both the original monitoring information and the additional MISP intelligence context.

3.2 Integration, interoperability and deployment

3.2.1 Connectivity Test

As explained above, the main update concerns the integration layer, where MISP has been connected with the Wazuh/Suricata monitoring flow so that alerts and network security events can be correlated against MISP IoCs and external threat intelligence feeds, enriching local detections with context such as known malicious infrastructure, malware associations, threat events, source organisations and threat levels.

Using the following request, one can use this request test API call from the Wazuh/MISP integration side to the MISP server to test functionality into the deployed MISP tool, under our project's GitLab repository. It checks whether the MISP server is reachable and whether the provided API key is valid. It does not send Wazuh or Suricata alerts yet, it is a connectivity and authentication validation request only.

Request:

```
curl -k -s -H "Authorization: <key>" https://misp-core.100.64.12.11.nip.io:30443/servers/getVersion.json
```

Response:

```
{
  "version":"2.5.36",
  "perm_sync":true,
  "perm_sighting":true,
  "perm_galaxy_editor":true,
  "perm_analyst_data":true,
  "uuid":"45e1ff11-c79d-4aa7-84f9-822d11fac035"
}
```

© CYberSynchrony, 2024

3.2.2 Feed Database

MISP runs its own MariaDB inside the `misp-core` pod. It's populated by external threat-intel feeds/data sources — in our case only the Computer Incident Response Center Luxembourg (CIRCL) OSINT Feed (Figure 2). MISP pulls IoCs from CIRCL on a schedule and stores them in its local MariaDB and when Wazuh queries MISP (**/attributes/restSearch**), MISP answers from that local DB (no further API call out to CIRCL). IoCs are our artefacts that can indicate malicious activity, such as suspicious IP addresses, malicious domains, phishing URLs, or malware file hashes. In CYberSynchrony, IoCs are checked against MISP to determine whether they are already associated with known malware, phishing campaigns, botnets, spam infrastructure, ransomware activity, or other threat intelligence events.

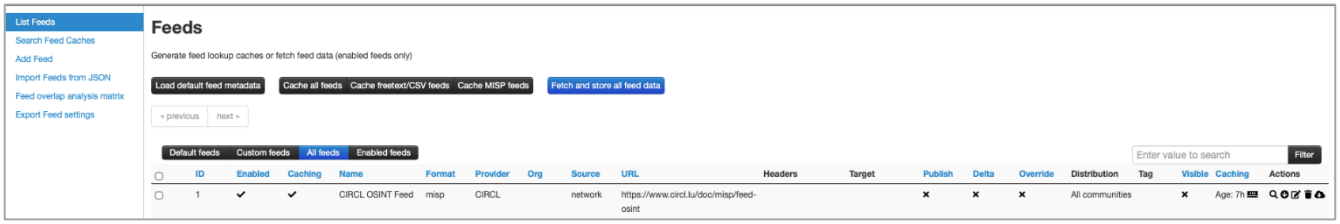

Figure 2. MISP feeds enabled

For the project's scope, IANUS extended the threat intelligence integration by aligning MISP with the outputs of T4.4, namely the CYberSynchrony Vulnerability Database, led by UPRC. This database operates as a structured repository of high-priority vulnerabilities, misconfigurations and cyber risks relevant to the project, currently containing 16,604 Critical and High severity entries collected from sources such as NVD and CISA KEV. The database includes 113 CISA Known Exploited Vulnerabilities, which are prioritised due to their confirmed exploitation in real-world attacks (as of 15 June 2026, when the database was received by IANUS). Each record includes structured fields such as CVE ID, vulnerability description, publication date, severity, CVSS score, CWE, affected vendor and product, mitigation action, source and status. The integration is based primarily on the CVE_ID field, which allows MISP to correlate vulnerability intelligence with threat events, indicators, affected technologies and security findings produced by other CYberSynchrony tools. Through this alignment with UPRC, IANUS enhanced the MISP knowledge base beyond traditional IoC enrichment by incorporating vulnerability-centric intelligence from T4.4, enabling analysts to connect malicious infrastructure, observed alerts and affected software products with known exploitable weaknesses. This improves the operational value of MISP within CYberSynchrony, as it supports more accurate prioritisation of threats, better contextualisation of Wazuh/Suricata alerts, and stronger evidence for downstream risk assessment, incident analysis and response planning during the pilot deployments.

3.2.3 Integration efforts

The integration logic is that Wazuh and Suricata detect suspicious activity first, and MISP is then used to enrich that activity with threat intelligence from its vulnerability databases. More specifically:

- Suricata first monitors network traffic on the on-premise VM and produces raw network security events in eve.json, including fields such as ***src_ip, dest_ip***, ***src_port***, ***dest_port***, ***proto, app_proto***, ***alert.signature_id***, ***alert.signature***, ***HTTP hostname/URL***, ***packet counts*** and ***byte counts***. These raw Suricata events are then collected by the Wazuh agent and sent to the Wazuh Manager in the cloud, where they are written into the unified alert file: ***/var/ossec/logs/alerts/alerts.json*** inside the Wazuh Manager pod, also available on the host at ***/data/wazuh-manager/logs/alerts/alerts.json***.
- Wazuh does not simply copy the raw Suricata event. It wraps and enriches it with its own metadata. For example, a Suricata alert becomes a Wazuh alert with a rule object, agent object, manager object, decoder, location, timestamp and internal Wazuh alert ID. The original Suricata fields remain preserved under the data object. So the downstream tools do not read raw Suricata telemetry only; they read

© CYberSynchrony, 2024

Wazuh-enriched JSON alerts that include both the original network event and Wazuh's classification metadata. Wazuh alerts already contain the fields needed for MISP lookups (***data.src_ip***, ***data.dest_ip***, ***data.alert.signature_id***, ***data.http.hostname***, etc.). Given any alert, we can extract an IoC and query MISP.

- The MISP integration then uses the indicators already present inside the Wazuh alert. For example, from a Suricata/Wazuh alert we can extract fields such as ***data.src_ip***, ***data.dest_ip***, ***data.http.hostname***, ***data.http.url***, file hashes if present, or other IoC-like values. These values are queried against MISP using its API. The documentation proves this feasibility: MISP is reachable and authenticated, has around 170,000 IoCs loaded from the CIRCL OSINT feed, and a lookup for IP 182.65.96.201 returned a structured MISP match identifying it as ip-src under "Potential SpamBots (2016-03-15)" from CIRCL.

The actual enrichment logic can be seen in the graph below (Figure 3).

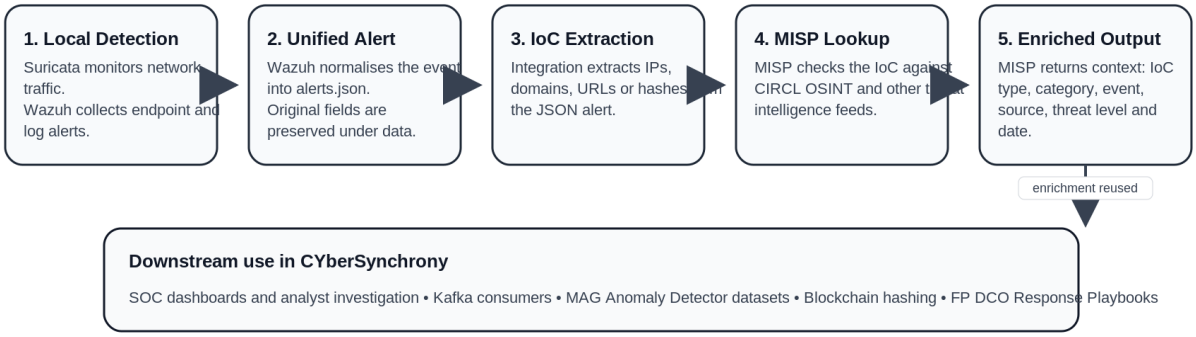

Figure 3. MISP Intelligence Flow

Table 3: Wazuh alerts used for MISP lookup

Source field in Wazuh alert	Used for MISP lookup
data.src_ip	Source IP lookup
data.dest_ip	Destination IP lookup
data.http.hostname	Domain / hostname lookup
data.http.url	URL lookup
File hash fields, when present	Malware / file hash lookup
data.alert.signature_id / data.alert.signature	Contextual correlation / rule reference

The MISP integration uses the indicators already present inside the Wazuh alert. For example, from a Suricata/Wazuh alert, the system can extract values such as ***data.src_ip***, ***data.dest_ip***, ***data.http.hostname***, ***data.http.url***, file hashes where present, or other IoC-like values. These values are queried against MISP using its API. The feasibility of this integration has been demonstrated: MISP is reachable and authenticated, it contains approximately 170,000 IoCs from the CIRCL OSINT feed, and a lookup for IP 182.65.96.201 returned a structured MISP match identifying it as ip-src under "Potential SpamBots (2016-03-15)" from CIRCL.

The actual enrichment logic is as follows: Wazuh/Suricata detect an event; the integration extracts IoCs from the alert JSON; MISP checks whether these IoCs are known in its threat intelligence database; MISP returns contextual information such as IoC type, category, event title, source organisation, threat level, and publication date; and this enrichment result can then be used by the Wazuh/MISP integration workflow and downstream consumers. Depending on deployment configuration, the enriched information may be displayed to analysts,

© CYberSynchrony, 2024

stored as enrichment metadata, forwarded to Kafka, or used by other modules such as MAG's Anomaly Detector pipeline or FP's DCO Response Playbooks.

3.2.4 Added value of integration activities

The difference between the alert before and after MISP enrichment is summarised below.

Table 4: Before and After MISP enrichment

Stage	Information for SOC Analyst
Before enrichment	Wazuh/Suricata alert with timestamp, source/destination IP, protocol, port, signature, severity, and affected agent.
After enrichment	Original alert plus MISP context, including matched IoC type, related threat event, source organisation, threat level, date, category, and known malicious association.

This means that an alert is no longer treated only as an isolated local event. It becomes part of a wider threat intelligence picture, enabling better prioritisation, investigation, and response planning.

As for the added value of this integration, let's take for example the case where we detect a Suricata-only alert (no MISP integration so far). The JSON output looked like this, **BEFORE** the nitration efforts:


```
{
  "timestamp": "2026-05-07T06:29:31.011+0000",
  "rule": {
    "level": 3,
    "description": "Suricata: Alert - ET INFO DYNAMIC_DNS Query to nip .io Do
main",
    "id": "86601",
    "groups": ["ids", "suricata"]
  },
  "agent": { "id": "002", "name": "cybersync4_100.64.12.14" },
  "data": {
    "src_ip": "100.64.12.11",
    "dest_ip": "192.168.1.1",
    "proto": "UDP",
    "alert": {
      "signature": "ET INFO DYNAMIC_DNS Query to nip .io Domain",
      "category": "Potentially Bad Traffic",
      "severity": "2"
    },
    "dns": { "query": [{ "rrname": "wazuhdashboard.100.64.12.11.nip.io", "rrt
ype": "A" }] }
  },
  "location": "/var/log/suricata/eve.json"
}
```


The example below is the outcome **AFTER** the MISP integration and enhancement.


```
{
  "timestamp": "2026-05-07T07:19:48.612+0000",
  "rule": {
```


© CYberSynchrony, 2024

```
    "level": 11,
    "description": "MISP: suspicious IP 182.65.96.201 - Network activity",
    "id": "100101",
    "groups": ["misp", "threat_intel", "misp_match", "network"]
  },
  "agent": { "id": "002", "name": "cybersync4_100.64.12.14", "ip": "100.64.12
.14" },
  "data": {
    "integration": "misp",
    "misp": {
      "event_id": "38",
      "category": "Network activity",
      "type": "ip-src",
      "value": "182.65.96.201",
      "comment": "Imported via the freetext import.",
      "permalink": "https://misp-
core.100.64.12.11.nip.io:30443/events/view/38"
    }
  },
  "location": "misp"
}
```


As we can observe, **the added value is that the MISP integration adds external threat intelligence context that was not available in the Suricata-only alert**. Before enrichment, the alert only showed a local network event with technical details such as IPs, protocol, DNS query, signature and severity, while after enrichment it is correlated with a known MISP IoC and converted into a higher-severity threat intelligence alert. This gives the SOC analyst the matched IoC type, category, event ID, indicator value and direct link to the relevant MISP event, allowing faster validation and investigation. As a result, the integration improves prioritisation, traceability and response planning by turning isolated Wazuh/Suricata detections into actionable intelligence-backed security findings.

In general, when Wazuh queries MISP for a specific IoC, it gets back structured JSON as follows:


```
{
  "id": "9418",
  "category": "Network activity",
  "type": "ip-src",
  "value": "182.65.96.201",
  "comment": "Imported via the freetext import.",
  "uuid": "56e7fc04-3aa0-4a38-b125-4cfc950d210f",
  "timestamp": "1458043908",
  "Event": {
    "id": "38",
    "info": "Potential SpamBots (2016-03-15)",
    "date": "2016-03-15",
    "Org":  { "name": "ADMIN" },
    "Orgc": { "name": "CIRCL" },
    "ThreatLevel": { "id": "3", "name": "Undefined" }
  }
}
```


© CYberSynchrony, 2024

The explanation for each field is presented below:

Table 5: MISP IOC Categories

JSON Field	Explanation
value	The IoC itself (182.65.96.201)
type	Category (ip-src, ip-dst, md5, domain, yara, snort, etc.)
category	Higher-level grouping (Network activity, Payload delivery, etc.)
Event.info	Human-readable title of the threat report this IoC belongs to
Event.date	When the report was published
Event.Orgc.name	Contributing organization (e.g., CIRCL — Luxembourg CERT)
Event.ThreatLevel	Undefined / Low / Medium / High
timestamp	When this attribute was added/updated

For example, lets assume that the on-premise Kubernetes cluster – via the Wazuh/Suricata – integration picks up this network alert and sends it to our server:

```
src_ip = 143.204.194.100
dest_ip = 100.64.12.14
alert.signature = GPL ATTACK_RESPONSE id check returned root
http.hostname = testmynids.org
```

The following actions will occur:

1. Wazuh wraps this into a structured alert under rule 86601, with groups ["ids", "suricata"].
2. From there, an integration script can extract ***src_ip***, ***dest_ip or http.hostname*** and query MISP.

If MISP finds a match, the alert is no longer just “Suricata discovered suspicious traffic.” It becomes “Suricata saw suspicious traffic, Wazuh classified it as an IDS alert, and MISP confirms whether the IP/domain/hash is linked to known spam, malware, phishing, ransomware or other threat intelligence.”

From a system perspective, Suricata provides network-level detection, Wazuh centralises, normalises, and enriches the event into the unified alert stream, and MISP adds external threat intelligence context. **For CYberSynchrony, the added value is the correlation between local evidence and external IoC intelligence**. This enriched context can later be used by other tools, such as MAG’s Anomaly Detector pipeline, Kafka consumers, blockchain hashing components, and FP’s DCO Response Playbooks, to support better actionable insights and recommendations.

3.2.5 Dashboards and Analytics

To illustrate the above, if we peak into the Wazuh Dashboard, the integration results and the IoC extra correlation is depicted through the following Figure 4:

© CYberSynchrony, 2024

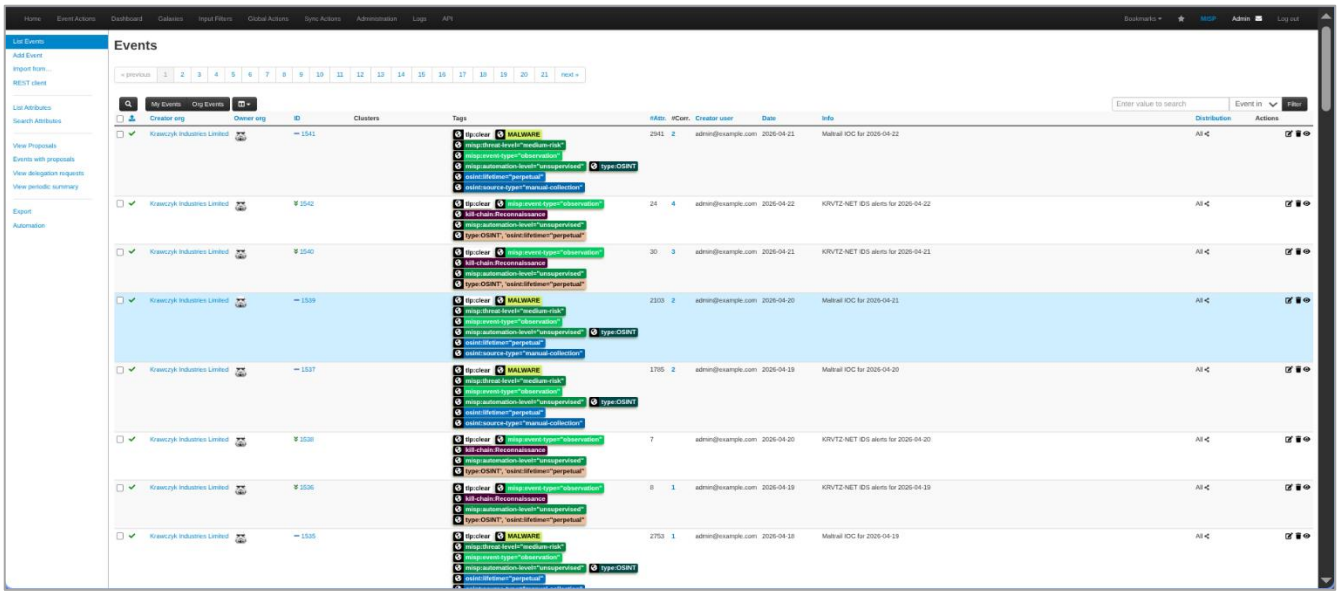

Figure 4. MISP Dashboard - Events

MISP displays information about various organizations and the events they've published, along with tags that aid in contextualizing the event's contents, e.g. what kind of incident was observed, from what country a suspicious IP address derived from, what kind of tool was used etc. (Figure 5).

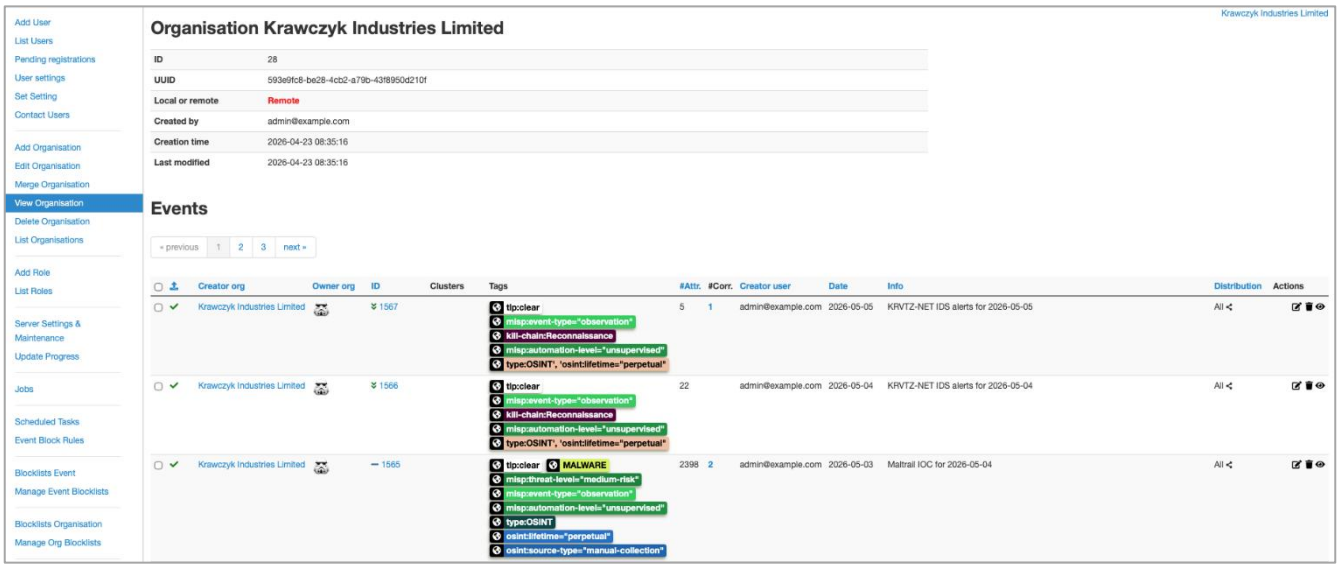

Figure 5. MISP Dashboard – View Organization

MISP incident investigation shows the event detail for "KRVTZ-NET IDS alerts for 2026-05-05" (Figure 6). It shows creator org, tags ***(`tlp:clear` , `kill-chain:Reconnaissance`***), distribution (All communities), and 5 attributes (Figure 7). The right sidebar shows a related event (1556) that shares at least one IoC - automatic correlation.

© CYberSynchrony, 2024

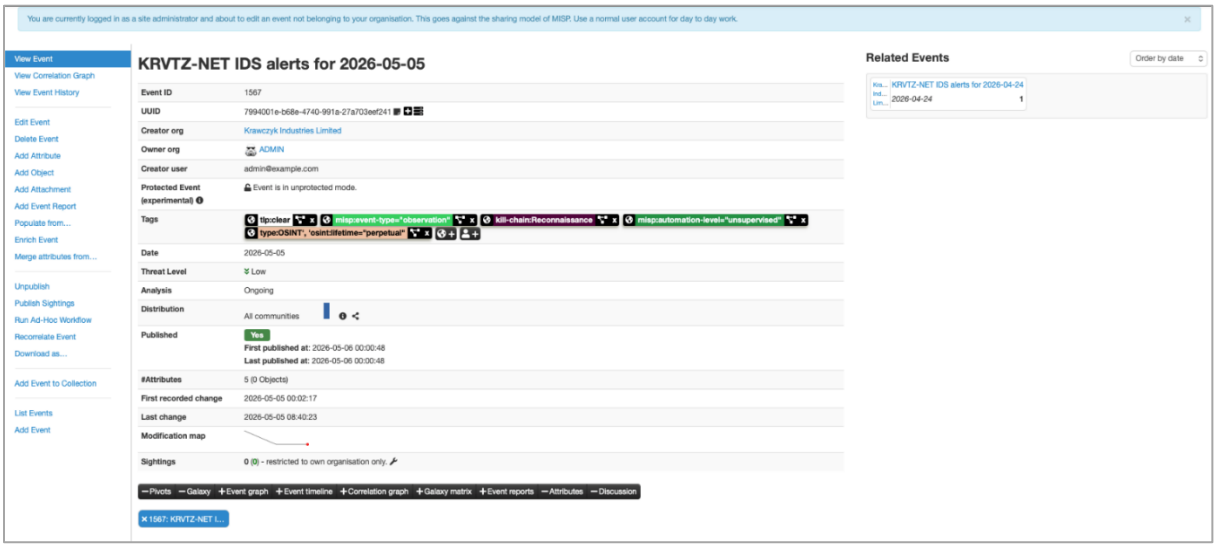

Figure 6. MISP Dashboard - Investigate incident

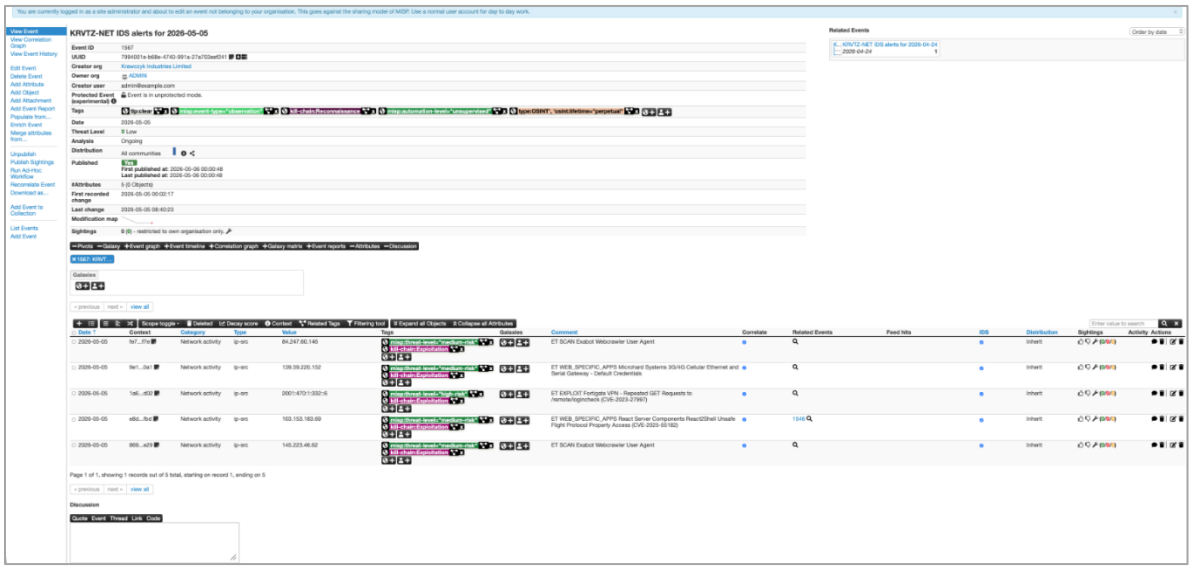

Figure 7. MISP Dashboard – Event attributes

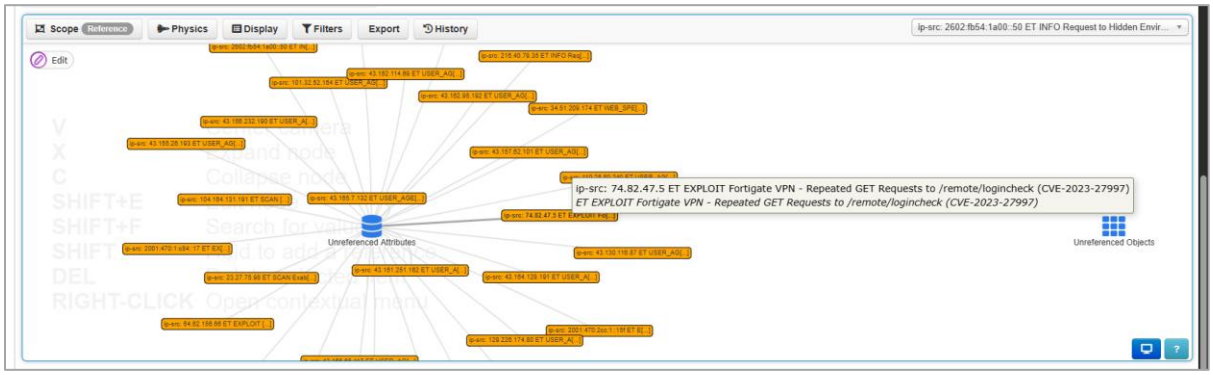

Figure 8: MISP Dashboard - Event Graph

The Event Graph in Figure 8 displays the visualization of objects and attributes in the event, with a description given on each object and attribute.

© CYberSynchrony, 2024

Figure 9: MISP Dashboard - Adding Tags

As shown in Figure 9, in this newly created event, tags are being added (‘***tlp:amber’***, ‘***OSINT’***, ‘***Cobalt Strike Beacon’***, ‘***Country:Pakistan’***) which provide help with threat intelligence classification, with Global tags being shared with other MISP instances and Local tags being mainly used for internal workflow operations. Figure 10 shows the option for adding a Galaxy Cluster to the event.

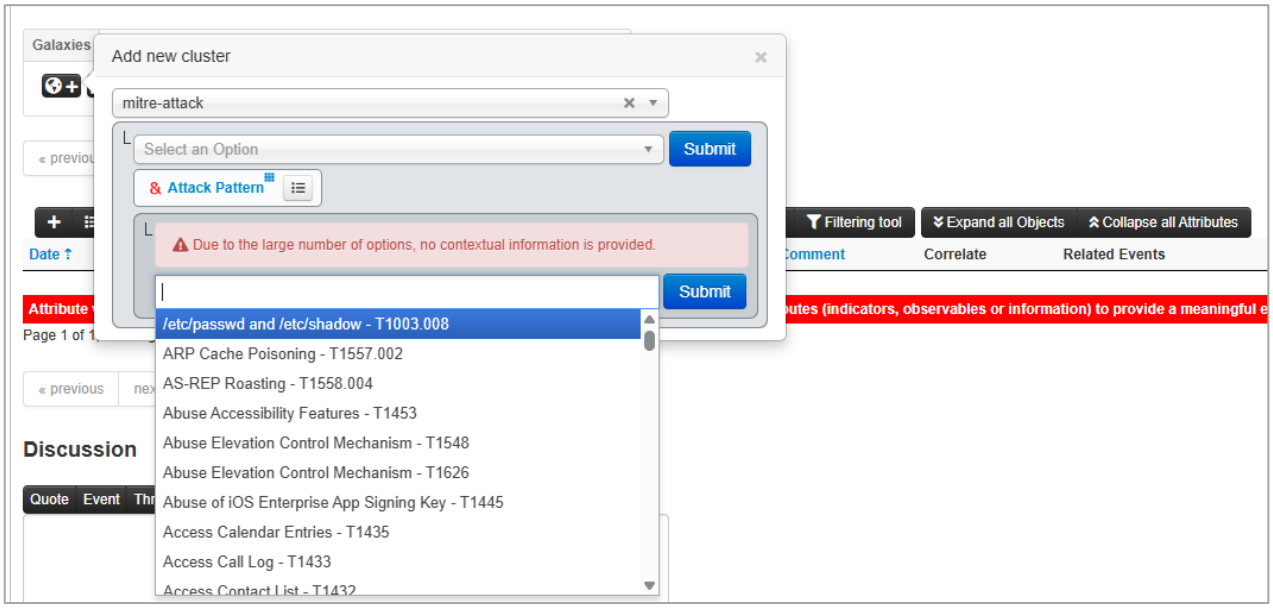

Figure 10: MISP Dashboard - Add Galaxy Cluster

© CYberSynchrony, 2024

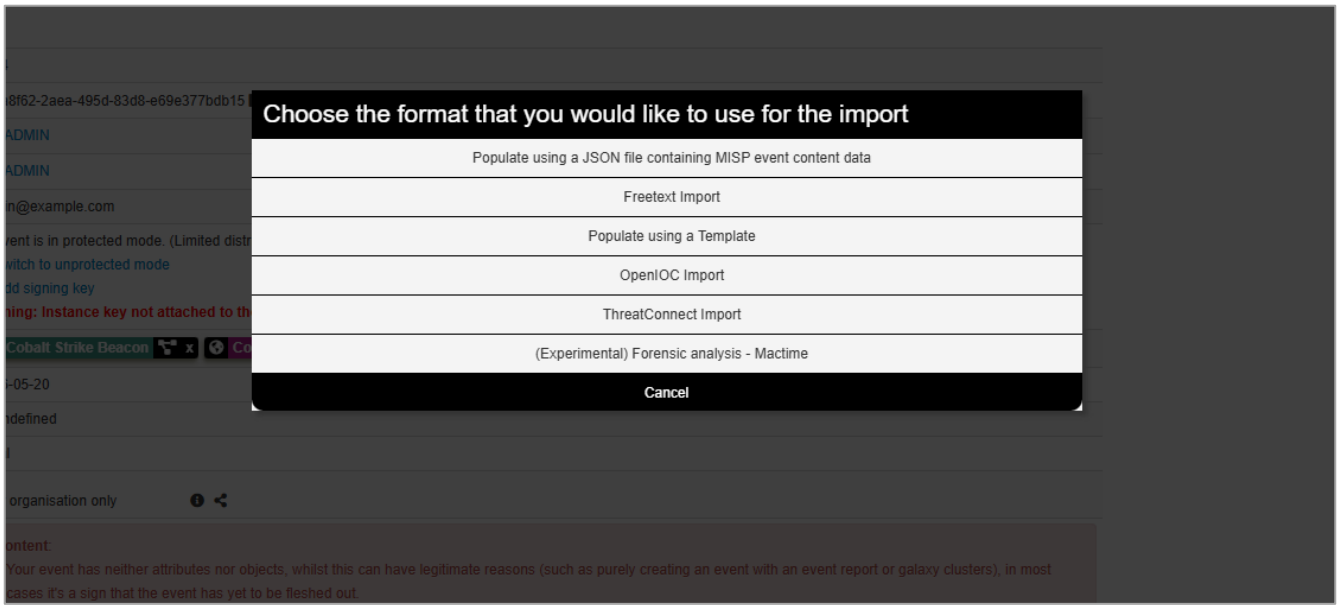

Figure 11: Populate from - Dropdown Menu

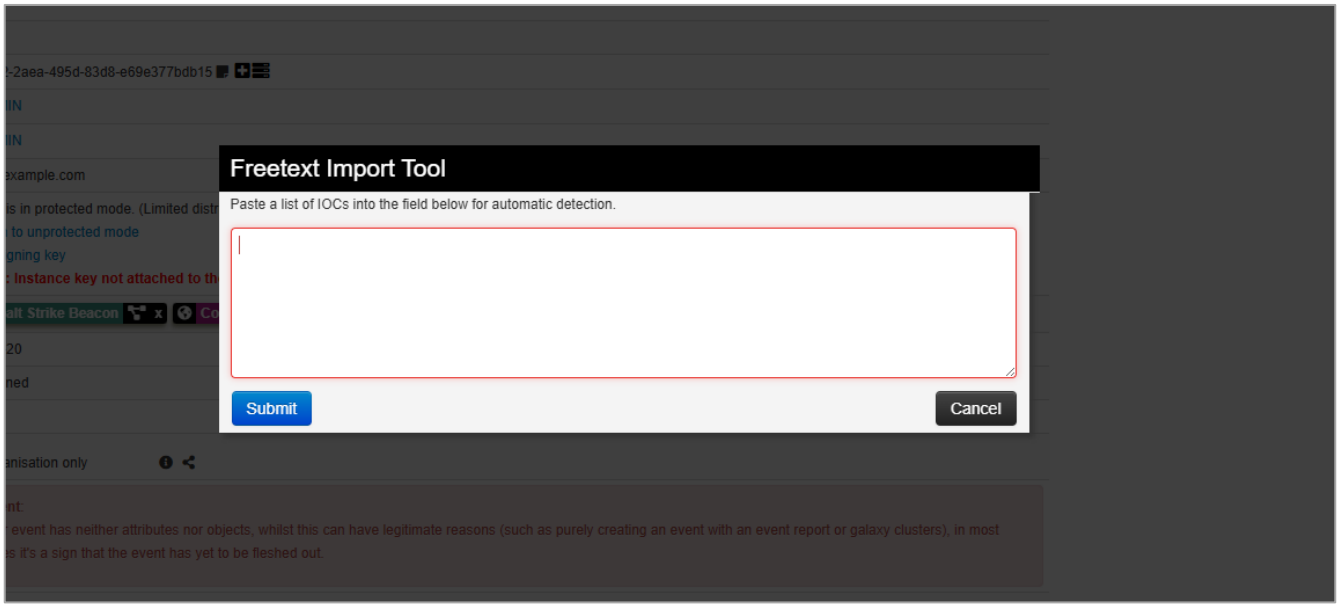

Figure 12: Populate from - Freetext Import Tool

MISP also provides automated tools for importing and structuring data during event population. As shown in Figure 11, these tools can be accessed through the Populate from drop-down menu. One of the available options is the Freetext Import Tool, illustrated in Figure 12, which parses text-based reports and automatically converts the extracted information into MISP reports, objects, attributes, and other relevant event elements.

Wazuh can extract IP addresses from security alerts and check them against MISP using the MISP API endpoint. For example, using the MISP API endpoint **/attributes/restSearch**, Wazuh communicated 5 IPs found in its logs and asked MISP for attributes (IOCs) – like IP address, domain, URL, file hash or email address.

Table 6: Wazuh – MISP API communication

Value (IP Address)	Type	Comment
84.247.60.146	ip-src	ET SCAN Exabot Webcrawler

139.59.220.152	ip-src	ET WEB_SPECIFIC_APPS Microhard Systems
2001:470:1:332::6	ip-src	ET EXPLOIT Fortigate VPN — CVE-2023-27997
103.153.183.69	ip-src	ET WEB_SPECIFIC_APPS React2Shell — CVE-2025-55182
145.223.46.62	ip-src	ET SCAN Exabot Webcrawler

Although Wazuh is documented in detail under Deliverables D4.1 and D4.3, the MISP integration can also be validated through the Wazuh Dashboard, where enriched threat intelligence events are stored and displayed in the event database. As shown in Figure 13, the Wazuh Dashboard (*do note that the Wazuh server holds all network traffic and alert logs*) includes a MISP-generated alert with the rule description "**MISP: suspicious IP 182.65.96.201 - Network activity**", a higher severity level, and a dedicated MISP rule ID. This confirms our integration and that MISP enrichment is not only performed at API level, but is also propagated back into the operational monitoring interface used by SOC analysts.

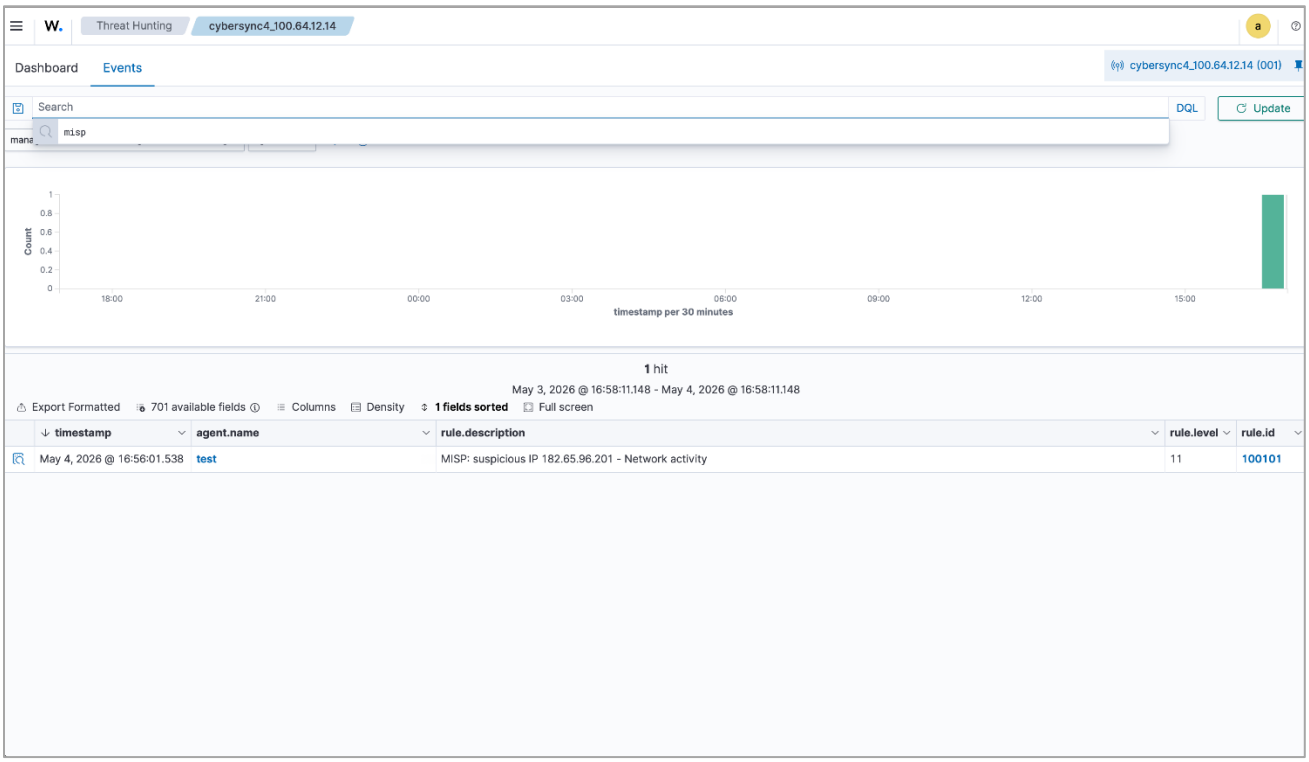

Figure 13. Wazuh dashboard with MISP metadata

4 Kitsune Search

4.1 Updates since v1

While the technical characteristics, endpoints, and specifications of the RESTful APIs of Kitsune Search are comprehensively detailed in the first version of this deliverable (D3.1), the current updated version (D3.3) integrates the specific work performed to achieve integration and interoperability with the CYberSynchrony platform.

A major achievement in this phase is the deployment of a dedicated web-based graphical user interface (GUI) integrated into the project's centralized dashboard, allowing analysts to move beyond raw JSON data processing.

This front-end provides authorized users with direct access to Kitsune Search services, transforming backend intelligence into intuitive visual modules for tracking threat actors, managing the digital attack surface, and triaging underground information leaks.

By embedding these capabilities into the broader platform architecture, CYberSynchrony ensures that specialized threat intelligence, such as botnet exposures and dark web mentions, is presented in an actionable, human-readable format that significantly enhances real-time threat detection and risk evaluation workflows.

Additional details can be found in the *Annex I: PoC and Prototypes (Kitsune Search)* section.

4.2 Integration, interoperability and deployment

In the v2 of the *Enhanced Threat Intelligence System (D3.3)*, the Kitsune Search tool transitions from the baseline standalone component established in v1 to an integrated API-driven service and a web front-end within the unified CYberSynchrony platform architecture.

Integration is always facilitated through RESTful APIs exchanging data in JSON format, allowing the platform to perform on-demand, targeted queries regarding data leaks, underground forum mentions and botnet infections related to specific internal assets.

This interoperability ensures that Kitsune Search enriches findings from global intelligence platforms like MISP and local monitoring agents like Wazuh and Suricata, providing critical dark web and entity-based context directly to the centralized dashboard

Graphical User Interface

While the foundational version of Kitsune Search focused on providing backend RESTful APIs for machine-to-machine communication, version 2 introduces a web-based front-end integrated into the CYberSynchrony platform.

This interface allows SOC analysts and operators to move beyond raw JSON data and interact with threat intelligence through intuitive dashboards and visual analytics.

Once logged in, the user can navigate through several specialized views that transform Kitsune’s extensive database of threat actors, botnets, and data leaks into actionable security insights.

This consolidated integration directly supports several project performance metrics:

- **KPI 1.1 (Number of Threat Assessments Carried Out):** by documenting concrete threats such as leaked credentials or dark web mentions affecting pilot assets, Kitsune Search ensures that these documented cases are automatically captured in the Threat Assessment Register.
- **KPI 1.3 (Threat Intelligence Platform Effectiveness)**: Kitsune’s ability to provide immediate context to security alerts improves the platform's capacity to correctly detect, classify, and route threats to DCO playbooks, contributing to the project's target of a >85% detection and mitigation rate.

© CYberSynchrony, 2024

- **KPI 3.7 (Risk Mitigation Index)**: the interoperability between Kitsune and the risk scoring engines enables pilot leaders (UPRC, AMC, EGL) to perform more accurate "what if" scenario analyses, facilitating the required 20% reduction in identified risks through data-driven mitigation strategies.

© CYberSynchrony, 2024

5 Unalterable Forensic Techniques adaptation (UFT)

5.1 Updates since v1

Version v1 of this deliverable (D3.1) introduced the Unalterable Forensic Techniques (UFT) adaptation through a Python component called the Data Preparation Module (DPM). The DPM operated on batches of logs: for each batch it computed a single SHA-256 hash, registered that hash as a transaction on the project's permissioned Hyperledger Fabric ledger, and uploaded the original batch to cloud storage. Verification later re-read the batch, recomputed its SHA-256 and queried the ledger. This v1 design correctly captured the chain-of-custody objective but, in operational use, exposed three limitations that v2 directly addresses.

First, batch granularity is too coarse: a tampering event affecting even a single alert inside a batch invalidates the entire batch's hash and forces a full re-investigation, while telling the analyst nothing about which event was altered. Second, a plain SHA-256 hash over an aggregated file does not bind the integrity proof to any particular producer; if the cloud-side store is reachable by more than one writer, an attacker who can also recompute and re-anchor a fresh hash defeats the control silently. Third, the v1 design implicitly assumed that the same byte sequence is available on the agent and on the cloud side, which — as detailed in the next subsection — does not hold for several Wazuh data sources.

Version v2 therefore evolves the DPM into a pair of Hyperledger-Fabric-aware components: a signer sidecar on the on-prem node (cybersync4) that applies a keyed Message Authentication Code per alert, and a Wazuh custom integration on the cloud node (cybersync1) that verifies that MAC before the alert is anchored to the ledger. The Hyperledger Fabric ledger and the cloud-storage retention model from v1 are retained unchanged; what changes is the cryptographic primitive, the granularity at which it is applied, and the placement of the integrity check within the Wazuh pipeline.

Since version v1 of D3.3, the Unalterable Forensic Techniques (UFT) adaptation has been re-scoped from a static file-fingerprint approach to a per-alert, keyed message-authentication design. The driver for this change was the realisation that the Wazuh data plane is, by design, append-only and continuously mutating: the Suricata eve.json file grows without bound, the Wazuh Manager wraps every received event with additional metadata on arrival, and several agent-native data sources (Security Configuration Assessment, syscollector, rootcheck) never materialise as a flat file on the agent at all. A single hash over a file therefore cannot serve as a meaningful integrity proof end-to-end.

The revised design protects the smallest meaningful unit — the individual alert — by attaching a keyed Message Authentication Code (MAC) at the agent and verifying it at the manager. The MAC is then anchored into the project's permissioned Hyperledger Fabric ledger, producing a tamper-evident chain that bridges the on-prem agent (cybersync4) and the cloud manager (cybersync1).

The most significant changes with respect to v1 are summarised below:

- Cryptographic primitive: introduces HMAC-SHA256 (secret_key, alert) as the per-alert integrity proof. HMAC is the standard, purpose-built message-authentication construction (RFC 2104, FIPS 198-1) and — unlike a naive keyed concatenation of the form SHA-256(key || data) over a Merkle-Damgard hash — is immune to length-extension attacks. The v1 design used an unkeyed SHA-256 hash per log batch (via the DPM) and therefore provided no producer authentication; v2 binds every individual alert to its producing agent through a keyed MAC.
- Granularity and producer binding: v1 anchored one hash per batch with no notion of which agent produced the data; v2 anchors one MAC per alert, computed with a per-agent secret_key provisioned at enrollment. Compromise of one agent invalidates only that agent’s signed stream; all other agents remain cryptographically independent.
- Wire format and key confidentiality: only the MAC output travels on the wire, in a dedicated “hmac” field of the alert. The secret_key never leaves the agent host nor the manager’s secret store, and no key material is ever embedded in the alert payload under any field name.

© CYberSynchrony, 2024

Constant-time verification: manager-side comparison uses hmac.compare_digest to avoid timing side-channels that could leak information about the expected MAC value.
Operationalisation: the signer is delivered as a sidecar inside the Suricata pod on cybersync4, and the verifier as a Wazuh custom integration on cybersync1. Both consume the secret_key from a Kubernetes Secret rather than from a ConfigMap or an in-image file.

5.2 Design rationale and threat model

This section documents, in deployment-ready detail, how forensic integrity is established for every alert produced on the on-prem node (cybersync4) and verified on the cloud node (cybersync1) before being anchored into the Hyperledger Fabric ledger. The pipeline spans Suricata, the Wazuh Agent, the Wazuh Manager, a custom integration module and the Distributed Hyperledger Fabric network.

5.2.1 Goal and threat model

The objective is to guarantee that an alert observed by an analyst on the cloud-side Wazuh Dashboard, or anchored in the Hyperledger Fabric ledger, is bit-for-bit the same alert that was produced on the on-prem agent – or, if it is not, that the discrepancy is deterministically detected and surfaced as a high-severity security event.

The in-scope adversary model includes: (i) a network attacker with the ability to intercept and modify traffic on the WAN segment between cybersync4 and cybersync1 in spite of the existing Wazuh transport encryption (defence-in-depth assumption); (ii) a compromised intermediate host or proxy able to replay, drop or alter individual alerts; and (iii) an insider with read-only access to the cloud-side archives who attempts to rewrite history. Out of scope is a full compromise of the on-prem agent host with root access to the secret_key file, which would by definition allow the attacker to mint valid MACs; that scenario is covered by the orthogonal SCA / FIM controls of Wazuh itself.

5.2.2 Why a simple file fingerprint is insufficient

An earlier design considered computing a single SHA-256 over /var/log/suricata/eve.json on the agent and re-computing it on the manager. Three properties of the Wazuh data plane make this approach unworkable in practice:

- eve.json is continuously appended to by Suricata at line rate (thousands of events per minute under our test workload). Any hash taken over the file is stale before it can be transmitted.
- The Wazuh Manager does not store agent payloads verbatim. On arrival each event is wrapped in manager-side metadata (agent.id, agent.ip, manager.name, decoder, rule, geolocation, MITRE tags, ingestion timestamps) before being written to /var/ossec/logs/archives/archives.json. A file-level hash taken on the agent therefore cannot match the cloud copy by construction.
- Several agent-native data producers — notably the Security Configuration Assessment (SCA) module and syscollector — never write their output to a flat file before forwarding. They place binary IPC messages directly into /var/ossec/queue/ subdirectories that are consumed by ossec-agentd and forwarded over the encrypted channel. There is no file on disk to hash.

The conclusion is that integrity must be enforced at the granularity of the individual JSON alert, not at the granularity of a log file, and it must be applied at the point in the agent pipeline where every alert is already in a uniform, readable form.

5.2.3 Unification point: where Wazuh and Suricata streams converge

Inside the agent the two relevant streams - network alerts generated by Suricata and host-based events generated by Wazuh's native modules - converge at the Wazuh internal queue under /var/ossec/queue/. This queue is, however, a set of binary UNIX-socket-backed IPC structures, not a textual JSON stream. It cannot be tailed by a generic tool and is not a safe surface to modify, since it is owned and re-opened by the Wazuh daemons.

© CYberSynchrony, 2024

The chosen architectural workaround is to read from the source producers directly rather than from the queue: a sidecar process (sign_alerts.py) tails /var/log/suricata/eve.json for network alerts and reads the JSON spool files produced by the SCA and syscollector modules. It merges them into a single readable file, /var/log/unified_alerts.json, which becomes the canonical pre-signing surface. This file is the last point in the pipeline at which agent and manager can agree on a deterministic byte representation of an alert, and it is therefore the natural anchor for the MAC computation.

5.3 Cryptographic design

5.3.1 Cryptographic primitive: HMAC-SHA256

Each alert is authenticated with HMAC-SHA256 as standardised in RFC 2104 and FIPS 198-1. The primitive is invoked as

mac = HMAC-SHA256(key = secret_key, message = canonical_json(alert_without_hmac_field))

Three properties motivate the choice of HMAC over a bespoke keyed-hash construction:

- Length-extension resistance: SHA-256 is a Merkle–Damgård hash and the naïve construction SHA-256(key || message) is vulnerable to length-extension attacks. HMAC's nested-keying construction is provably secure against these attacks under standard assumptions on the underlying compression function.
- Specification clarity and review: HMAC-SHA256 is implemented in the Python standard library (hmac.new(..., digestmod=hashlib.sha256)) and is available in every platform we deploy on. Operators and auditors do not need to reason about a custom construction.
- Performance: HMAC-SHA256 over a typical alert (1–4 KiB) costs on the order of microseconds and is negligible relative to the cost of JSON encoding and disk I/O. We measured no observable impact on Suricata's drop counters at our test traffic rates.

Canonicalisation is performed with json.dumps(alert, sort_keys=True, separators=(',', ':'), ensure_ascii=False).encode('utf-8'). Sorting the keys and removing insignificant whitespace ensures that the agent and the manager hash the same byte sequence even if intermediate JSON libraries reorder fields. The hmac field, if present in the input dictionary, is removed before canonicalisation so that signing and verification operate on the identical pre-MAC representation.

5.3.2 Per-agent secret_key lifecycle

Each Wazuh agent is provisioned with its own 256-bit secret_key (32 bytes from os.urandom) at enrollment time. The manager generates the key, stores it locally in a Kubernetes Secret (wazuh-agent-keys), and delivers it to the agent once over the existing encrypted enrollment channel. On the agent the key is written to /var/ossec/etc/integrity.key with 0600 permissions, owned by root, and is never logged, never echoed to standard output, and never embedded in container images.

Key rotation is supported by re-running the enrollment procedure. During a rotation window the manager temporarily accepts MACs computed under either the old or the new key for a configurable grace period (default: 5 minutes) so that in-flight alerts are not falsely flagged as tampered.

Per-agent isolation is a deliberate property of the design: because the manager maintains a key per agent.id, the compromise of one agent's host invalidates only that agent's signed stream. The verifier rejects any alert whose agent.id does not match a key it holds, preventing a compromised agent from impersonating another.

© CYberSynchrony, 2024

5.4 End-to-end alert pipeline

5.4.1 Agent-side signing flow (cybersync4)

The signing sidecar runs in the same Kubernetes pod as Suricata and shares the eve.json volume. Its operational loop is:

- Read one newline-delimited JSON record from the unified source (Suricata eve.json or a Wazuh queue spool file).
- Parse the record into a Python dict; remove any pre-existing hmac field defensively.
- Canonicalise: payload_bytes = json.dumps(alert, sort_keys=True, separators=(',',':'), ensure_ascii=False).encode('utf-8').
- Compute mac = hmac.new(secret_key, payload_bytes, hashlib.sha256).hexdigest().
- Attach the MAC: alert['hmac'] = mac.
- Append the signed record to /var/log/signed_alerts.json, which is the file declared to the Wazuh Agent's logcollector as a json-formatted log source.

From this point on the standard Wazuh transport (encrypted TCP to manager port 30114) carries the signed alert unchanged. The signer never speaks to the network and never opens an outbound socket; this minimises its attack surface and keeps it auditable.

5.4.2 Manager-side verification flow (cybersync1)

On the cloud node the Wazuh Manager receives each signed alert and writes it to /var/ossec/logs/archives/archives.json as it would for any other event. A custom integration (custom-verifyintegrity.py, declared in ossec.conf under <integration>) is invoked on every archives.json write and performs the following sequence:

- Look up the per-agent secret_key in the Kubernetes Secret wazuh-agent-keys, indexed by the alert's agent.id field. If no key is found the alert is treated as untrusted and Wazuh rule 100202 (level 12) is raised.
- Pop the received MAC: mac_received = alert.pop('hmac').
- Recompute the canonical byte representation of the remaining alert and compute mac_expected with the agent's secret_key.
- Compare in constant time with hmac.compare_digest(mac_received, mac_expected).
- On match: tag the event with integrity.verified = true and forward a record to the Hyperledger Fabric anchoring service. On mismatch: raise Wazuh rule 100200 (level 14) “Integrity violation: HMAC mismatch”, suppress the forward to Hyperledger Fabric, and leave the original (suspected-tampered) event in archives for forensic review.

Because verification is gated on the agent.id, the Wazuh Manager additionally rejects alerts whose claimed agent.id does not correspond to a key it knows; this prevents trivial spoofing of the agent.id field by a network attacker.

5.4.3 Tamper-detection rules and SOC visibility

Three Wazuh rules support the integrity pipeline; they are shipped in k8s/wazuh/integrations/integrity_rules.xml:

- Rule 100200 (level 14, groups: integrity, tamper) — HMAC mismatch. Triggers an immediate notification to the SOC channel and is excluded from any auto-suppression filters.
- Rule 100201 (level 3, groups: integrity, verified) — HMAC verified. Suppressed from the dashboard by default but indexed for audit so that the absence of verification events on a given agent is itself detectable.

© CYberSynchrony, 2024

- Rule 100202 (level 12, groups: integrity, missing_key) — Alert received from an agent with no registered secret_key. Indicates either an enrollment failure or an attempt to inject events for an unknown agent.

On the Wazuh Dashboard a dedicated Discover saved-search (rule.groups: integrity) and a small visualisation set (tamper-events-over-time, top-agents-with-mismatches) provide the SOC analyst with a single pane of glass on the health of the integrity pipeline.

5.4.4 Anchoring verified alerts into Hyperledger Fabric

Only alerts whose MAC verified successfully are forwarded to the Hyperledger Fabric anchoring service. The forwarded record contains the alert's logical identifier (agent.id, timestamp, rule.id, an event UUID derived from the canonical bytes) and the MAC itself; the full alert payload is retained in Wazuh archives and is not duplicated into the ledger. This keeps the ledger compact while preserving its evidentiary value: any later dispute over the content of an alert can be settled by re-canonicalising the archived alert, recomputing its MAC, and comparing against the anchored value.

Anchoring is performed in micro-batches (default: 100 records or five seconds, whichever comes first) using the existing chaincode interface described in the blockchain sub-project. Each batch produces a single ledger transaction whose Merkle root is logged back into Wazuh as rule 100203 (level 3), providing a bidirectional link between the SIEM and the ledger.

5.5 Implementation, security and deployment

5.5.1 Implementation artefacts

The following files implement the design and are versioned in the project repository:

- suricata/sign_alerts.py — Python signer sidecar. Tails the Suricata eve.json and Wazuh queue spool files, performs canonicalisation and HMAC-SHA256, writes /var/log/signed_alerts.json. Loads secret_key from /var/ossec/etc/integrity.key (mode 0600).
- k8s/suricata/deployment.yaml — Kubernetes manifest that adds the signer as a sidecar container alongside Suricata, mounts the integrity-key Kubernetes Secret, and configures a shared emptyDir for the unified and signed log files.
- k8s/wazuh/integrations/custom-verifyintegrity.py — Manager-side verifier invoked through Wazuh's <integration> hook. Loads the per-agent key map from the wazuh-agent-keys Kubernetes Secret, performs the constant-time HMAC comparison and emits the appropriate decoded event.
- k8s/wazuh/integrations/integrity_rules.xml — Wazuh rule definitions for the three integrity rules (100200, 100201, 100202) and for the Hyperledger Fabric anchoring confirmation rule (100203).
- k8s/wazuh/secrets/wazuh-agent-keys.yaml — template for the Kubernetes Secret that stores one base64-encoded secret_key per agent.id. Real key material is injected at deployment time via the project's secret-management workflow and is never committed to the repository.

© CYberSynchrony, 2024

5.5.2 Security analysis and residual risks

Under the threat model declared above the design provides the following guarantees. (i) Any single-alert modification between the agent's signed_alerts.json and the manager's archives.json is detected with overwhelming probability (collision probability ≈ 2^-256). (ii) Replay of an unmodified alert is detected at the application layer by the existing Wazuh deduplication on (agent.id, timestamp, rule.id); the Hyperledger Fabric anchor provides an additional ordering proof. (iii) Spoofing the agent.id of another agent is prevented by per-agent keys.

Residual risks that are explicitly accepted in v2 and tracked for future iterations: (a) silent drop of alerts on the agent before signing is not detected by the integrity layer alone and is covered instead by Wazuh's keep-alive monitoring; (b) full root compromise of the agent host allows an attacker to mint valid MACs and is mitigated only by host-hardening and by anomaly detection on the alert volume and timing; (c) the secret_key rotation procedure requires a brief grace window during which both old and new keys are accepted, which is a deliberate trade-off between operational continuity and cryptographic hygiene.

5.5.3 Interoperability and deployment status

The integrity pipeline interoperates transparently with the existing IANUS components: MISP enrichment runs before verification on the manager (so MISP-tagged alerts are equally authenticated), and the Kitsune Search front-end consumes verified alerts only, by filtering on integrity.verified: true. No changes are required in the dashboard layer or in the MISP integration scripts. At the time of writing the cryptographic design is complete and reviewed; the signer, the verifier and the Wazuh rules are implemented and validated on the staging cluster against synthetic and real alert traffic.

© CYberSynchrony, 2024

6 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. By the time of writing this deliverable, no pilot has begun producing KPI measurements. Pilot operations start from M25, with the first are expected during M25 and M26.

6.1 KPI1.3: Threat Intelligence Platform Effectiveness

KPI ID	**KPI Name**	**Owner**
1.3	Threat Intelligence Platform Effectiveness	IANUS
Target		
Achieve threat detection & mitigation rate >85% out of 100.		
Measurement Methodology		
Components involved	**Participating partners & Role in KPI Measurement**	**Evidence collection process & validation approach**
MISP	IANUS uses MISP to test whether threat intelligence indicators, such as malicious IPs, domains, URLs, hashes, malware-related indicators and ransomware-related IoCs, are detected, enriched and correlated with pilot-relevant assets or events.	MISP event exports, IoC lookup results, matched IoC records, API responses, screenshots, event IDs and analyst notes.
Wazuh	IANUS uses Wazuh to test whether endpoint, authentication, privilege escalation, configuration, rootcheck, vulnerability or other security events are detected, classified and made visible to analysts.	Wazuh alert IDs, JSON alert exports, dashboard screenshots, rule IDs, rule descriptions, severity levels, affected agent/asset, timestamps and analyst acknowledgement.
Suricata	IANUS uses Suricata to test whether network-based attacks or suspicious traffic, such as command-and-control traffic, malware communication, abnormal connections or IDS signature matches, are detected and forwarded into the Wazuh monitoring flow.	Suricata alerts, eve.json extracts, Wazuh-normalised Suricata alerts, source/destination IPs, ports, protocols, signatures, rule IDs, severity and screenshots.
Anomaly Detector IDS	MAG uses the Anomaly Detector IDS to test whether anomalous traffic conditions are detected compared to the learned baseline and whether the alert reaches the analyst with enough information to interpret it.	Anomaly alert payloads, anomaly score, reconstruction error, affected time window, contributing flow features, protocol, ports, byte counts, packet counts, flow duration and peer pairs.

© CYberSynchrony, 2024

Kitsune Risk/Monitoring	MER uses Kitsune Search / Monitoring to test whether suspicious external references, exposed information, suspicious IP/domain intelligence or abnormal risk signals are identified and provided as useful context for threat assessment.	Kitsune reports, screenshots, monitoring outputs, search results, external references, affected pilot/use case and analyst notes.
DCO Playbooks	FP uses the DCO component to verify whether detected and classified threats can be routed to a relevant recommendation, mitigation path or response playbook.	Playbook IDs, recommendation records, screenshots, response workflow logs, analyst acknowledgement and closure notes.

Methodology Description

The Threat Effectiveness Test Log will be used as the evidence register for this KPI. Each row in the log records one simulated or confirmed threat event and tracks whether the platform handled it successfully from detection to analyst action. The log should include the date, pilot/use case, test or confirmed threat event, tools involved, expected outcome, actual outcome, detection result, enrichment result, classification result, routing result, analyst acknowledgement, and links to supporting evidence such as alerts, reports, screenshots, JSON exports or dashboard extracts. The platform will be tested with known, simulated or confirmed cyber events during pilot and validation activities. For each event, the responsible partner checks whether the platform detected the event, added useful context, classified it correctly, routed it to a recommendation or playbook, and enabled the analyst to acknowledge and interpret it. Since CYberSynchrony follows a recommendation-based response approach, “mitigation” does not necessarily mean automatic blocking. For this KPI, mitigation means that the platform generates a valid recommendation or response playbook and that the analyst acknowledges it as relevant and actionable. A threat event is considered successfully handled when the platform completes the full operational chain: detection, enrichment where applicable, classification, routing to a recommendation or playbook, and analyst acknowledgement. The effectiveness rate is calculated as:

Correctly handled threat events / total tested or confirmed threat events x 100

For example, if 40 threat events are tested and 35 are correctly detected, enriched where relevant, classified, routed to recommendations or playbooks, and acknowledged by the analyst, then the platform effectiveness rate is: 35 / 40 x 100 = 87.5%

Additional test scenarios may include ransomware-related IoCs imported through MISP, brute-force login attempts detected by Wazuh, malware communication identified by Suricata, suspicious domain or IP intelligence provided by Kitsune, anomalous network scans detected by the Anomaly Detector, unusual data exfiltration patterns, lateral movement simulations, privilege escalation events, phishing-related indicators, or multi-stage attack scenarios where several tools contribute evidence. For example, for the Anomaly Detector IDS, effectiveness is measured through its ability to detect predefined anomalous traffic conditions during validation. These conditions produce Suricata flow features that deviate from the trained baseline in terms of traffic volume, timing, protocol distribution, peer-pair behaviour or data transfer patterns. For each tested condition, MAG records whether the Anomaly Detector raised an alert above the configured reconstruction-error threshold, whether the alert reached the analyst interface with anomaly score and contributing features, and whether the analyst could interpret and route it. The Anomaly Detector contribution is calculated as correctly flagged anomalous conditions divided by total tested anomalous conditions, multiplied by 100. Attack-type classification is performed by the analyst using the contributing flow features and correlated evidence, not automatically by the model. Threat events may also be evaluated

© CYberSynchrony, 2024

through correlation between multiple tools. For example, if Suricata detects suspicious malware communication, MISP confirms that the destination IP is linked to known malicious infrastructure, and Wazuh shows related activity on the same pilot asset, the platform output is considered stronger because several tools support the same threat interpretation.

KPI Progress Status			
Current Value/Target	**Date of Measurement**	**Status**	**Next Actions** (if required)
0/>85%	31/05/2026	On Track	Create the Threat Effectiveness Test Log, begin recording validated findings once pilot data collection starts.

Challenges, Deviations and Corrective Actions
A key challenge for this KPI is to ensure that platform effectiveness is measured against meaningful test events, not against simple alert volume. To address this, each event will be counted as successfully handled only if it is detected, enriched or contextualised where applicable, classified, routed to a recommendation or playbook, and acknowledged by the analyst. Another challenge is that different tools may contribute different parts of the process, so the Threat Effectiveness Test Log must clearly record which tool detected the event, which tool enriched it, and which component produced the recommendation or playbook.

6.2 KPI1.5: Threat Intelligence Platform Security

KPI ID	**KPI Name**	**Owner**
1.5	Threat Intelligence Platform Security	MAG
Target		
Reduce vulnerability remediation time to ≤15 days.		
Measurement Methodology		
Components involved	**Participating partners & Role in KPI Measurement**	**Evidence collection process & validation approach**
ASA (Attack Surface Assessment)	FP uses ASA to probe the public-facing surface of the deployed platform, identifying exposed assets, open services, CVEs, and attack-surface weaknesses that affect Threat Intelligence platform components.	ASA findings exports, asset inventory excerpts, exposed-service records, CVE listings, severity scores, screenshots, identification date and validated-closure date logged against each register row.

© CYberSynchrony, 2024

Wazuh	IANUS uses Wazuh agent telemetry, vulnerability module output, and rootcheck assessments to identify host-, endpoint- and configuration-level weaknesses on the systems that run MISP, Kitsune Search, and other Threat Intelligence platform components.	Wazuh alert IDs, vulnerability module reports, configuration assessment exports, affected agent/asset, severity level, timestamp, dashboard screenshots, and analyst confirmation that the finding has been remediated.
Suricata	IANUS uses Suricata to capture network-level signals (exposed protocols, unexpected service banners, exploitation attempts) that confirm or validate vulnerabilities affecting platform components, providing supporting evidence rather than primary discovery.	Suricata alert records, eve.json extracts, rule IDs, source/destination IPs, severity, screenshots; attached as validation evidence to the corresponding register row.
MISP	IANUS uses MISP to enrich each recorded vulnerability with CVE details, IoC associations, threat-actor context, and prioritisation information that helps order remediation work.	MISP event exports, CVE lookup results, IoC cluster references, analyst notes explaining the relevance of the enrichment to the recorded vulnerability.

Methodology Description

A Platform Vulnerability Register will be maintened, where each confirmed vulnerability affecting a CYberSynchrony Threat Intelligence platform component is recorded as one row. A finding counts as one register entry only when the detection partner has reviewed it, confirmed that it affects a deployed component, assigned a severity, applied a remediation action, and validated the fix. Minimum evidence for each row will include vulnerability ID, affected component, owner, detection tool/method, severity, date identified, remediation action, date fixed, date validated, and an evidence link.

Primary detection sources are ASA (FP) for exposed-service and attack-surface findings, and Wazuh (IANUS) for host- and configuration-level findings. Suricata (IANUS) supplies validation evidence at network level, and MISP (IANUS) supplies CVE and IoC enrichment that supports prioritisation. The responsible component owner performs the remediation, and UPRC, AMC, and EGL may support validation in the pilot environments where the affected component is deployed.

Remediation time for a single entry is calculated as the date of validated closure minus the date of identification, expressed in calendar days. The KPI value is reported as the average remediation time across all closed rows in the register. The benchmark of ≤15 days is met when this average sits at or below the threshold for the rows closed within the reporting window. Where the same vulnerability is detected by more than one tool, the earliest identification date is used and the row records all contributing evidence.

KPI Progress Status

Current Value/Target	**Date of Measurement**	**Status**	**Next Actions** (if required)
0/15 days	31/05/2026	On Track	Establish the Platform Vulnerability Register and share with FP and IANUS. Agree on the validation protocol with

© CYberSynchrony, 2024

			UPRC, AMC, and EGL for pilot environments. Confirm with MAG the cadence of aggregation and reporting toward D8.2.

Challenges, Deviations and Corrective Actions
A key challenge for this KPI is the absence of an initial baseline measurement: the benchmark phrases the target as a reduction, but no current-state value has been captured at consortium level. The register will therefore report the average remediation time directly against the 15-day target rather than as a relative reduction. There is also a risk of skew in simple averaging, since a single critical vulnerability that takes 60 days to validate can be hidden by several low-severity rows closed in two or three days. The register records severity per row, so the KPI can be re-cut by severity at M36 if the project officer asks for that view. Operational ownership is a third open item: MAG owns the KPI but does not run any of the detection tools, so the handover from TRUST, FP, and IANUS into the central register needs to be agreed in writing before scans begin.

© CYberSynchrony, 2024

7 Conclusions

D3.3 documents the second-phase integration state of the WP3 threat intelligence components at project month 25.

MISP is now operationally connected to the Wazuh/Suricata pipeline. Suricata alerts, enriched by Wazuh, trigger IoC lookups against a local database of roughly 170,000 indicators from the CIRCL OSINT feed. The connection has been validated: MISP is reachable and authenticated, lookups return structured threat context, and enriched alerts appear on the Wazuh Dashboard with dedicated rule IDs and elevated severity. What were previously isolated local detections now carry threat intelligence context that SOC analysts can act on without leaving the dashboard.

Kitsune Search's transition from a backend API to a dashboard-integrated front-end is complete. Analysts reach threat actor, botnet, and underground intelligence capabilities through the CYberSynchrony centralised interface, no raw JSON handling required. This feeds into KPI 1.1, 1.3, and 3.7.

The UFT component was redesigned in v2 to address three gaps in v1: coarse batch granularity, no producer authentication, and an assumption about byte-sequence consistency that Wazuh's append-only, metadata-wrapping data plane does not satisfy. Per-alert HMAC-SHA256 signing replaces batch hashing, providing producer binding at the individual alert level and per-agent cryptographic isolation for tamper detection. The signer and verifier are implemented and validated on the staging cluster; production rollout is tied to the Kafka ingestion task and gated on the same infrastructure change window.

Two items were open at submission: IANUS confirmation of UFT functionality on the on-premises node (cybersync4), which will be reflected in the final version, and the updated platform architecture diagram needed to complete Section 2.

This deliverable covers integration and interoperability specifications. Pilot results for UC1, UC2, and UC3 and their contribution to KPI measurement will be reported in WP8.

© CYberSynchrony, 2024

Annex I: PoC and Prototypes (Kitsune Search)

D3.3 integrates the specific work performed to achieve integration and interoperability with the CYberSynchrony platform. Deployment of a dedicated web-based graphical user interface (GUI) fully integrated into the project's centralized dashboard, allow SOC analysts to move beyond raw JSON data processing.

Integrated Kitsune Front-End and Intelligence Reach

This paragraph provides an overview of the processes and information accessible through the Kitsune Search front-end module integrated within the CYberSynchrony dashboard.

1. **Threat Landscape Overview (Homepage)**

The integrated homepage serves as the primary entry point for global and localized situational awareness.

It provides real-time statistics on the current global threat volume, including counts for tracked Threat Actors, Malware samples, and Indicators of Compromise.

Figure 14 shows the screen that allows you to access the services via the web GUI:

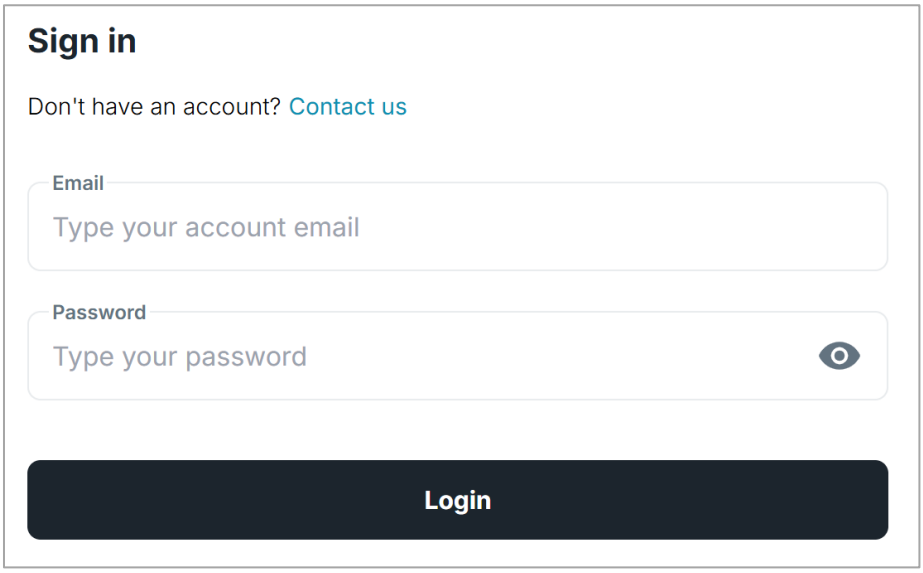

Figure 14: Kitsune Login

The main dashboard (Figure 15) prioritizes the most active threats and campaigns, allowing analysts to immediately see which actors (e.g., Turla, APT28) are most active and which sectors (e.g., Government, Technology) are currently under heavy attack.

© CYberSynchrony, 2024

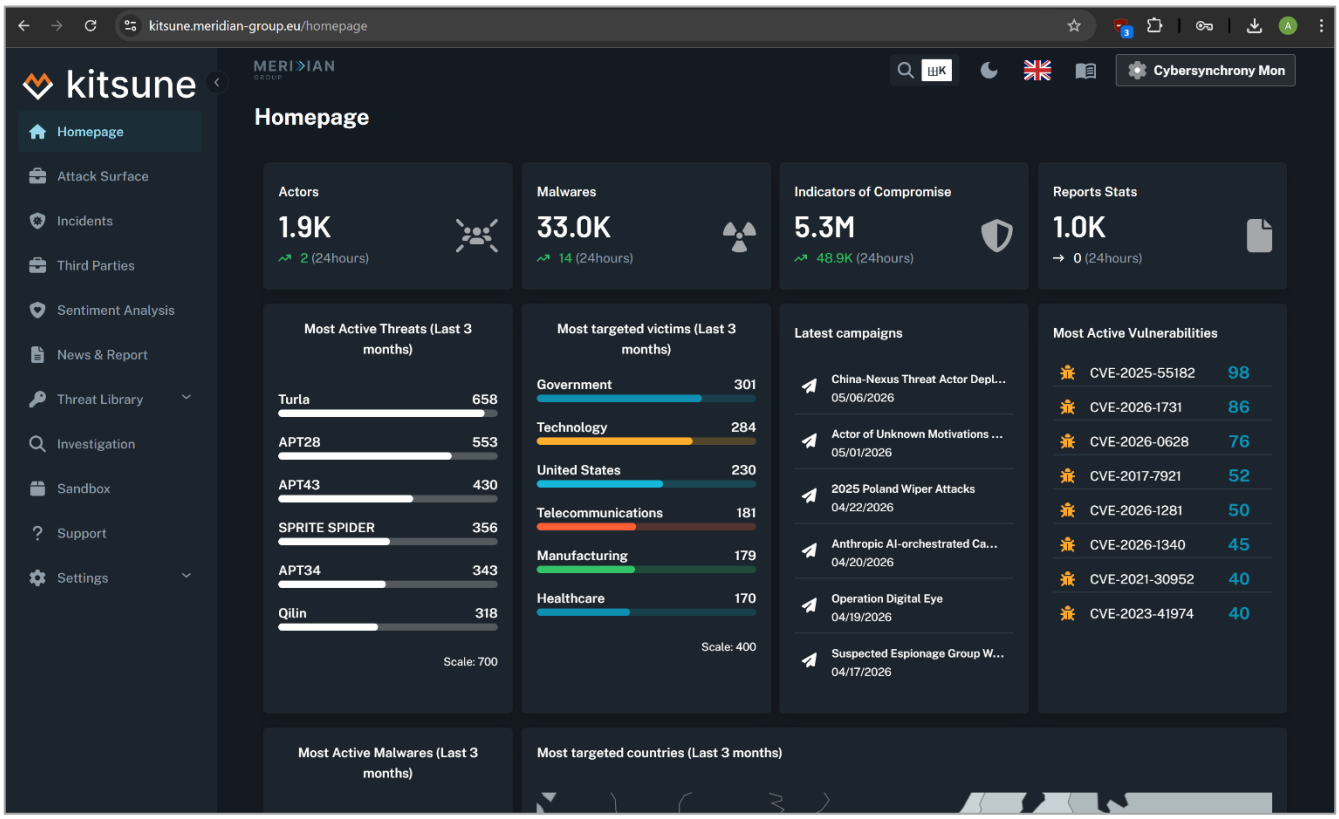

Figure 15: Kitsune Homepage

2. Underground and Darknet Search Interface

The Underground search portal (Figure 16) provides a unified search bar for executing on-demand queries across clandestine sources.

Analysts can pivot their investigations by selecting specific categories:

- **Darkweb Search (Generic)**: broad keyword searches across dark web forums.
- **Entity Search**: targeted lookups for compromised domains, IPs, or email addresses.
- **Bots & Leaks**: focused retrieval of data related to infostealer infections and archived data leak repositories.

© CYberSynchrony, 2024

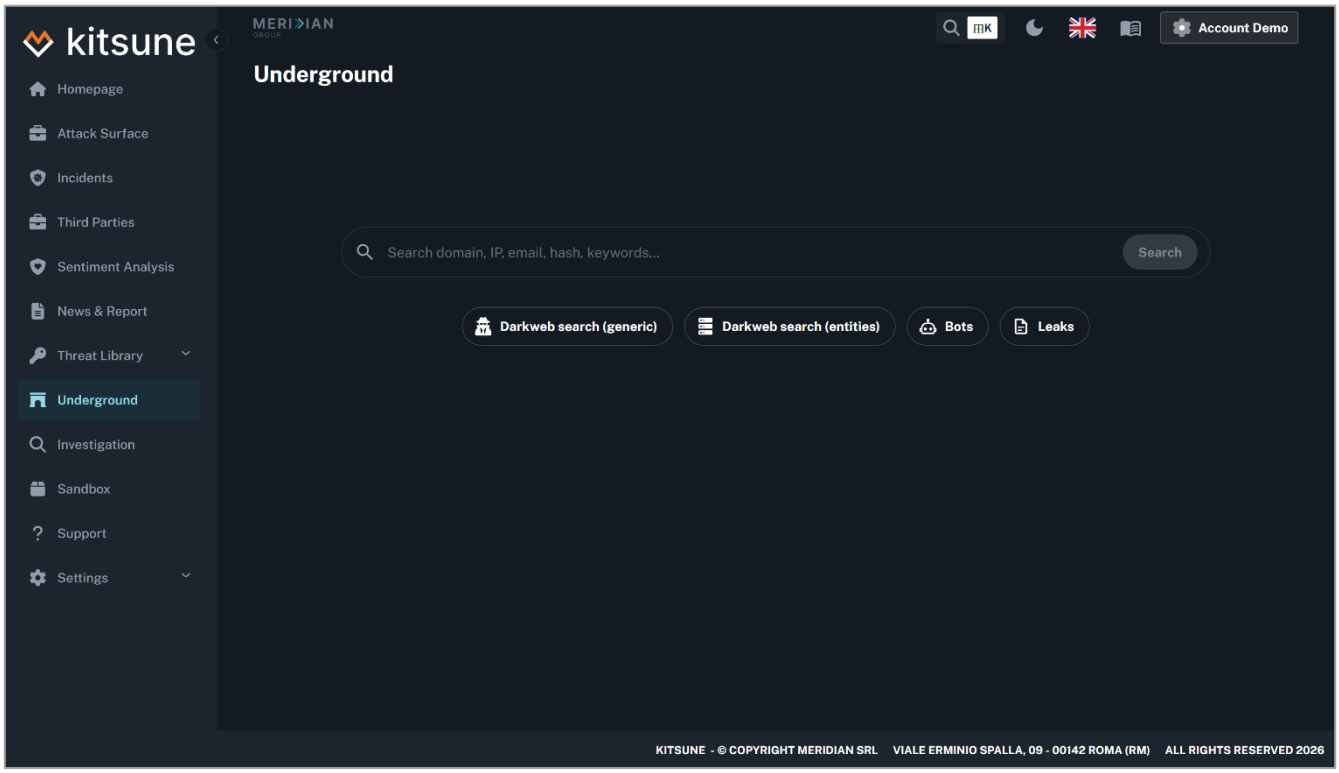

Figure 16: Kitsune Underground Search