

From Sandbox to Enforcement: Confidence-Qualified Threat Intelligence for Critical Infrastructure^{*}

Nikolaos Kekatos¹, Mihaela Curcă², Georgios Koutidis¹, Mihai Nena², Tom Nianios¹, Robert-Ştefan Şandru², Michael Ioannou³, and Charalambos Bratsas⁴

¹ Clone Systems, Cyprus, {nkekatos,gkoutidis,tnianios}@clone-systems.com

² Romanian National Cyber Security Directorate (DNSC), Romania,
{mihaela.curca,mihai.nena,robert.sandru}@dnsc.ro

³ Bolton Technologies, Cyprus, michael@bts.com.cy

⁴ International Hellenic University, Greece, cbratsas@ihu.gr

Abstract. Security operations centres and national incident-response teams defending critical infrastructure collect abundant threat data yet struggle to turn it into *actionable* intelligence. A malware sandbox produces detailed behavioural evidence, but as a large, unranked report whose confidence is unstated. We present CG-CTI, an operational pipeline that converts live sandbox output (CAPEv2) into STIX 2.1, correlates it in a knowledge graph with other critical-infrastructure sensors, and attaches to every intelligence object an explicit *confidence status* derived from provenance, cross-source corroboration, and observation durability. This status gates automated action: only corroborated intelligence is eligible for automated enforcement, while lower-confidence objects are routed to analyst review or kept as context. A grounded language-model stage then narrates the confidence-qualified evidence, where each statement either cites a supporting object or is marked unsupported, so fabricated references are removed before analyst review. We implement CG-CTI within the CYBERGUARD project, whose consortium includes Romania’s national cyber-security directorate, and evaluate it against the live sandbox on a labelled malware corpus, measuring conversion validity, indicator yield, technique coverage, corroboration, enforcement eligibility, latency, and summary grounding. CG-CTI turns fragmented sandbox output into corroborated, confidence-ranked, and auditable intelligence for critical-infrastructure defence.

Keywords: Cyber threat intelligence · Critical infrastructure protection · Malware sandbox analysis · Confidence qualification · STIX 2.1 · Evidence-grounded LLM summarisation

^{*} This is a preprint of an article published in the 21st International Conference on Critical Information Infrastructures Security (CRITIS 2026).

1 Introduction

Malware analysis is the systematic examination of malicious software to recover its behaviour, capabilities, and indicators of compromise [28]. It combines *static* inspection, *dynamic* detonation in an instrumented sandbox, and hybrid approaches, each with complementary blind spots: packing and obfuscation defeat static analysis, while anti-sandbox evasion and unexecuted code paths limit dynamic analysis [28,8]. Automated sandboxes operationalise dynamic analysis at scale: Cuckoo and its successor CAPEv2 [29], alongside commercial services, detonate samples and collect API-call traces, file-system and registry changes, network traffic, dropped files, and extracted configurations [28]. Malware analysis is thus a mature, well-tooled discipline; the difficulty this paper addresses is not *producing* behavioural evidence but *turning it into actionable, shareable threat intelligence* for defenders of critical infrastructure.

Security Operations Centres (SOCs) and national Computer Security Incident Response Teams (CSIRTs) defend entities that carry security and reporting obligations under the NIS2 Directive [10]. They do not suffer from a shortage of threat data; they suffer from a shortage of *actionable* threat data. A typical security-operations team receives on the order of several thousand alerts per day and ignores the majority of them [2], and threat data is routinely fragmented across tools and teams so that no single analyst sees the whole picture. The binding constraint is not visibility but *actionability*.

Malware sandboxes compound this problem. A dynamic-analysis system such as CAPEv2 detonates a suspicious sample and produces detailed behavioural evidence (process and network activity, dropped artefacts, signatures, and their ATT&CK mapping). This is the input a critical-infrastructure defender requires [21], yet it arrives as a large, sandbox-specific JSON report that is technically rich but not directly consumable by a SOC as prioritised, shareable intelligence. Compounding this, atomic indicators of compromise (IOCs) such as file hashes, IP addresses, and domains have a short operational lifetime, since adversaries rotate infrastructure once indicators are burned, whereas behavioural techniques are more durable and more expensive for an adversary to change.

This paper addresses the gap between high-fidelity sandbox output and actionable critical-infrastructure intelligence. We present CG-CTI, developed within the CYBERGUARD project (whose partners include the Romanian national cyber-security directorate, DNSC) and evaluated against a live CAPEv2 sandbox, which converts the output of that sandbox into STIX 2.1, correlates it in an OpenCTI knowledge graph with other sensors, attaches an explicit, evidence-derived *confidence status* to every intelligence object, and generates an evidence-grounded malware summary that an analyst validates before use. Unlike prior systems that generate cyber threat intelligence (CTI) for an analyst to read, CG-CTI attaches to each object a confidence status that gates automated enforcement.

Contributions. This paper makes four contributions.

- CG-CTI, an operational pipeline that ingests live CAPEv2 output as STIX 2.1 and correlates it with other critical-infrastructure sensors in an OpenCTI knowledge graph. Unlike prior CAPE→STIX converters, conversion is one stage of an evidence-to-enforcement lifecycle demonstrated up to the enforcement gate (Sect. 3).
- An *operational confidence model* that maps each object, via a single total function over provenance, cross-source corroboration, and observation durability, to an ordered status (*corroborated* > *attested* > *observed*) that *gates automated enforcement*: only corroborated intelligence is enforce-eligible. MISP records sightings and decays indicators and STIX 2.1 carries a confidence property; these describe confidence, whereas we make it a control-plane decision that gates enforcement (Sect. 4).
- *Evidence-grounded malware summaries* that invert the usual report→STIX direction of LLM-assisted threat intelligence: the model narrates already-converted, confidence-qualified STIX, with each statement either citing supporting evidence or explicitly marked unsupported; references to non-existent objects are dropped before delivery (12.1% at scale), so every surfaced claim is analyst-auditable (Sect. 5).
- An evaluation on a labelled corpus detonated through the deployed sandbox, quantifying conversion validity, yield, coverage, confidence-based enforcement selectivity, latency, and the grounding of the AI summaries (Sect. 7).

Novelty. CG-CTI combines in one pipeline what prior systems provide separately. It turns confidence from an analytic score into an *enforcement gate*: a per-object status, derived only from evidence the graph already records, deciding whether an object is auto-enforced, reviewed, or kept as context. And it inverts the direction of LLM-assisted CTI, narrating deterministically-converted, confidence-qualified STIX rather than extracting STIX from prose, so any statement citing evidence that does not exist is dropped before it reaches the analyst.

Structure. Section 2 motivates the problem; Sects. 3–5 present the architecture, the confidence model that gates enforcement, and the evidence-grounded summaries; Sects. 6 and 7 give the implementation and evaluation; and Sects. 8–11 position the work, discuss it, state limitations, and conclude.

2 Background and Motivation

Malware analysis and its intelligence yield. Malware analysis spans static inspection (headers, strings, hashes, disassembly), dynamic detonation in a sandbox, hybrid combinations of the two, and, increasingly, AI-assisted reverse engineering. Static analysis is thorough but slow and defeated by packing and obfuscation; dynamic analysis observes actual runtime behaviour and yields immediate operational indicators (contacted hosts, dropped payloads, persistence mechanisms), at the cost of missing dormant code paths and being susceptible to anti-sandbox evasion [28]. CAPEv2 [29] is an open dynamic-analysis system with kernel-level

API monitoring, network capture, memory forensics, and configuration extraction; its report is the primary intelligence source CG-CTI consumes.

CTI standards and platforms. Operationalising this output depends on shared representations. STIX 2.1 provides a machine-readable object model for observables (files, addresses, domains, URLs), higher-level constructs (malware, attack patterns, indicators), and their relationships [26]; TAXII defines how such intelligence is exchanged [27]. MITRE ATT&CK [35] supplies a common vocabulary of adversary tactics and techniques that lifts intelligence from atomic indicators to durable behaviour. Threat-intelligence platforms such as MISP [40] and OpenCTI [11] store, correlate, and visualise this intelligence; CG-CTI is built on OpenCTI and represents all sensor output as STIX 2.1.

From data to actionable intelligence. Integrating threat intelligence into SOC workflows (correlation, response playbooks, threat hunting) turns passive monitoring into intelligence-driven operations, but only if the intelligence is prioritised and trustworthy: poorly curated feeds overload a SIEM with false positives, and undifferentiated indicator streams worsen the very alert fatigue they are meant to relieve. The recurring lesson is that intelligence quality, context, and confidence matter more than raw quantity, and that behaviour-level intelligence outlasts atomic IOCs [41]. CG-CTI takes this lesson as its design premise: it makes confidence and durability explicit, per object, so that downstream SOC and CSIRT consumers can act on the intelligence that matters.

3 The CG-CTI Architecture

This section follows evidence through the deployed pipeline: ingested by purpose-built connectors, converted from CAPEv2 report to STIX 2.1, correlated in the OpenCTI graph, and served to analysts through a dashboard and API.

Ingestion: seven connectors into one knowledge base. CG-CTI is built on OpenCTI and fed by seven purpose-built STIX 2.1 connectors that group into three roles: a *malware-analysis* tier (the sandbox connector), a *detection* tier (host and network intrusion detection), and an *exposure-assessment* tier (vulnerability and code scanners). Each connector *normalises* before publishing: scanner findings arrive with their own identifier and severity schemes, and are mapped onto CVE/CWE and a common scale, so heterogeneous output lands in one correlatable graph (Fig. 1). The malware and detection tiers are this paper’s focus, since they carry the behavioural evidence a critical-infrastructure SOC acts on; the assessment tier situates that evidence within the broader attack-surface context and shares the same knowledge base, so no bespoke per-consumer integration is needed. The specific products behind each connector are listed in Sect. 6.

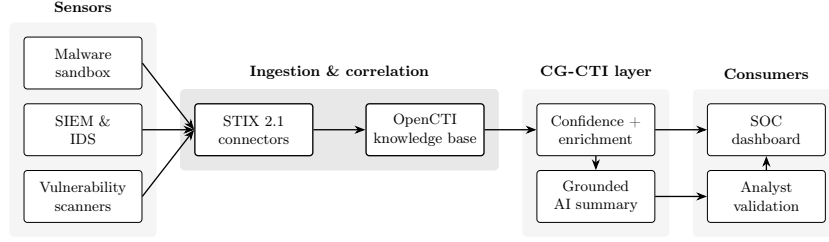


Fig. 1. The CG-CTI pipeline. STIX 2.1 connectors normalise heterogeneous sensor output into one OpenCTI knowledge graph, where the CG-CTI layer qualifies each object’s confidence, corroborates it against lightweight open-feed lookups, and generates an evidence-grounded AI summary that an analyst validates before it reaches the SOC dashboard. The sensors are a malware sandbox (CAPEv2), SIEM and host/network IDS, and vulnerability scanners; the SOC dashboard is the operator console.

Malware path: sandbox report to STIX. The sandbox connector retrieves each completed CAPEv2 report and normalises it into STIX 2.1 objects: the sample file and its hashes, a malware object typed from the reported family, the network indicators (contacted addresses, domains, and URLs) reconstructed from the sandbox telemetry, and attack-pattern objects mapping sandbox signatures to MITRE ATT&CK techniques. Relationships tie these to a source identity so provenance is preserved in the graph. The concrete STIX types and the report fields they draw from are given in Sect. 6.

Consumption: dashboard and API. A REST service exposes the knowledge base to a SOC-facing dashboard through endpoints for statistics, timelines, per-source alerts, malware families and their IOCs, and ATT&CK-technique coverage, so an analyst can move from a high-level indicator down to the underlying sandbox evidence within one consistent data model (Fig. 2). The current dashboard service is a pilot-grade read layer, and we return to its hardening in Sect. 10.

Correlation in the OpenCTI graph. Because every connector writes into one OpenCTI knowledge base, CG-CTI reuses OpenCTI’s *native* graph rather than building a separate store: identical observables emitted by different sensors (a hash seen by the sandbox and flagged by a detection sensor, a domain in both sandbox telemetry and a network alert) are merged into a single node, and four relationship types connect analyses into a navigable graph: **drops** (malware→file), **uses** (malware→attack-pattern) and **communicates-with** (malware→address, domain or URL) from the sandbox path, and **indicates** (indicator→object) from the detection and assessment tiers. This native correlation is the signal the confidence model’s corroboration axis reads (Sect. 4); the graph is not only a visualisation; it supplies the cross-source evidence on which confidence is computed.

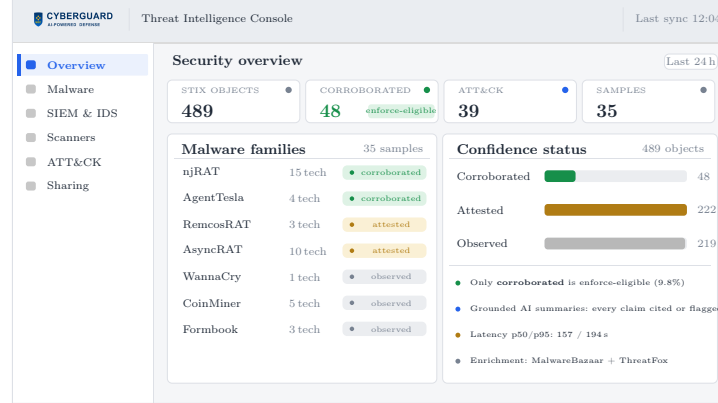


Fig. 2. The CG-CTI operator console in the CYBERGUARD testbed. It is drawn, not screenshotted, so it stays legible in print. All values are measured on the 35-sample corpus of Sect. 7. Each object carries a confidence status, and only **corroborated** objects are enforce-eligible.

4 An Operational Confidence Model: From Evidence to Enforcement

Raw sandbox and feed output arrives with heterogeneous scoring conventions: the sandbox emits a malware score, the vulnerability scanner a CVSS-derived *severity* score, and most connectors no comparable score at all. CG-CTI replaces these scattered heuristics with one model attaching an explicit confidence status to every object.

Three evidence axes. CG-CTI derives, for every intelligence object o , a confidence status from three axes the knowledge base already records. (i) *Provenance* $p(o)$ is a per-source base weight reflecting how directly the source witnessed the artefact: a dynamic behavioural observation from the sandbox (the sample actually contacted a host, dropped a file, or exhibited a signature) ranks above a static pattern match, which ranks above an external feed lookup. The ad-hoc per-connector scores already in the deployment are special cases of p : the sandbox’s malware score and the vulnerability connector’s CVSS-derived score ($conf = \min(10 \cdot CVSS, 100)$) are normalised into the provenance axis p , which CG-CTI unifies onto one scale. We use p as an ordinal rank, not a numeric threshold: *high* for a direct dynamic behavioural observation (the sample *exhibited* the behaviour) and *low* for a passive atomic observation (an address merely contacted, a hash of a dropped file). (ii) *Corroboration* $k(o)$ counts the sources beyond the originating sandbox that attest the same observable, another sensor already in the graph or an external-feed hit, so $k(o)=0$ means only the sandbox saw it. Because every connector writes into one OpenCTI graph, an identical hash, domain, or technique seen by the sandbox and a second source is automatically correlated. (iii) *Durability* $d(o)$

is the class of the observable: an *atomic* IOC (hash, IP, domain, URL), which adversaries rotate quickly, versus a *behavioural* object (a signature or ATT&CK technique), which is more durable. This axis operationalises the CTI survey’s [41] finding that behavioural techniques outlive atomic indicators, directly in the confidence model.

Status and rule. The three axes map to an ordered confidence status $C(o) \in \{\text{corroborated} \succ \text{attested} \succ \text{observed}\}$ by the rule

$$C(o) = \begin{cases} \text{corroborated} & k(o) \geq 1 \text{ (at least one source beyond the sandbox),} \\ \text{attested} & k(o) = 0 \text{ and } (d(o) = \text{behavioural or } p(o) = \text{high}), \\ \text{observed} & k(o) = 0 \text{ and } d(o) = \text{atomic and } p(o) = \text{low.} \end{cases}$$

Durability affects both initial qualification and subsequent ageing: behavioural evidence can remain **attested** without corroboration and retains that status, whereas an atomic IOC that is never corroborated ages out of the actionable set, so the model prefers durable evidence exactly as the survey recommends. Table 1 states what a SOC does with each level; only **corroborated** intelligence is eligible for automated enforcement, keeping low-confidence indicators out of the automated-enforcement path. The thresholds above are design defaults to be tuned against operator feedback (Sect. 10); the contribution is the explicit, per-object, evidence-derived status, not the specific cut points.

From evidence to enforcement. The model is a three-stage chain: the evidence axes form a per-object trust assessment, the rule maps that assessment to an ordered *confidence status*, and the status gates a *SOC action* (Table 1). Two properties keep the mapping principled rather than ad hoc. It is a total function: every object receives a status and none is left unranked. It is also monotone in each axis, since raising an object’s corroboration, durability, or provenance never lowers its status; the status is therefore an order-preserving image of the evidence, not a lookup table. Each cut point follows a stated principle: attestation by a corroborating source beyond the originating sandbox ($k \geq 1$) is the weakest evidence that licenses automated action, because a single sensor can be evaded; a durable behavioural technique or a directly witnessed behaviour merits analyst review even uncorroborated; and a lone passive atomic indicator is kept as context but never enforced.

Lightweight corroborating enrichment. The corroboration axis $k(o)$ is strengthened by a deliberately minimal enrichment step, kept small so its effect stays measurable. For each sandbox observable CG-CTI issues at most two open-feed lookups, a file-hash lookup against MalwareBazaar (family, tags) and an indicator lookup against ThreatFox (threat label) [1], reusing a single free abuse.ch key. A feed hit on the observable, a hash present in MalwareBazaar or an indicator listed in ThreatFox, counts as an external-feed attestation and promotes the object toward **corroborated**; the feed’s family or threat tags are recorded, but exact-label agreement is not required for promotion, because community labels are noisy. This

Table 1. Confidence status, the evidence that yields it, and the SOC action it gates.

Status	Evidence	SOC action
corroborated	a source beyond the sandbox attests it ($k \geq 1$)	enforce-eligible; shareable
attested	sandbox-only ($k=0$), behavioural or high-provenance	analyst-review queue
observed	sandbox-only ($k=0$), low-provenance atomic IOC	context only, not enforced

keeps enrichment cheap and its effect measurable. One caveat matters for reading the results, and we quantify it in Sect. 7: for a MalwareBazaar-drawn corpus a hash hit corroborates almost by construction, so the genuinely independent signals are ThreatFox network-indicator hits and cross-sensor corroboration.

Qualification is applied to every object as it enters the graph: the corroborating set K collects sources beyond the originating sandbox, an open-feed lookup may add to it for atomic observables, and the status follows Table 1.

5 Evidence-Grounded AI Malware Summaries

The confidence-qualified graph is machine-actionable, but an analyst or a CSIRT duty officer still needs a readable account of *what a sample does and why it matters*. Producing that by hand is slow; producing it with a large language model (LLM) applied to raw sandbox logs is fast but untrustworthy, because the model can assert behaviours the evidence does not support. CG-CTI takes a middle path: an *Evidence-Grounded Malware Summary* in which supported statements are tied to the STIX objects and sandbox evidence that justify them.

Grounding by construction. The AI stage never sees the raw CAPEv2 log. Instead, CG-CTI selects a compact evidence set from the confidence-qualified STIX graph, the **File**, **Malware**, network observables, and ATT&CK **AttackPatterns** for the sample, each associated with its derived confidence status, and passes only that structured evidence to the model. The model is instructed to write an executive summary, observed behaviours, ATT&CK techniques, IOCs, likely impact, and recommended mitigations, and to *cite, for each statement, the identifier(s) of the supporting STIX object(s)*. A statement with no supporting object is shown but marked unsupported, so the claim is visible as such instead of reading as fact; only a statement citing an identifier that does not exist is dropped. Because the evidence is already confidence-qualified, the summary inherits the same qualification: a behaviour asserted only from **observed** (uncorroborated) evidence is marked accordingly. Algorithm 1 makes the grounding check explicit: every generated statement is scored against the evidence set, and the report’s *grounding rate* (Q7) is the fraction of statements traceable to a valid evidence identifier. In the algorithm, `SELECTEVIDENCE` assembles the evidence set E from the

Algorithm 1 Evidence-grounded malware summary and grounding check.**Require:** sample s ; confidence-qualified STIX graph G ; self-hosted LLM M **Ensure:** report R ; grounding metrics

```

1:  $E \leftarrow \text{SELECTEVIDENCE}(G, s)$   $\triangleright$  evidence objects with associated  $C(\cdot)$ 
2: assign stable ids  $e_1, \dots, e_n$  to  $E$ ;  $ids \leftarrow \{e_1, \dots, e_n\}$ 
3:  $R \leftarrow M(\text{PROMPT}(E))$   $\triangleright$  JSON: statements, each with cited evidence ids
4:  $g \leftarrow f \leftarrow u \leftarrow 0$ 
5: for all statement  $t \in R$  do
6:   if  $\text{CITES}(t) = \emptyset$  then
7:      $f \leftarrow f + 1$   $\triangleright$  model-flagged unsupported
8:   else if  $\text{CITES}(t) \subseteq ids$  then
9:      $g \leftarrow g + 1$   $\triangleright$  grounded
10:  else
11:     $u \leftarrow u + 1$ ; drop  $t$   $\triangleright$  cites unknown id: hallucinated ref
12:  end if
13: end for
14:  $R \leftarrow \text{ANALYSTVALIDATE}(R)$ ; write  $R$  and status back to  $G$ 
15: return  $R, (g, f, u)$ 

```

graph, PROMPT renders it for the model, CITES returns the evidence identifiers a statement references, and ANALYSTVALIDATE is the human approval gate.

Analyst validation. The generated report is a draft for analyst review, not an autonomous verdict. An analyst validates, edits, or rejects it in the dashboard; the validated report and its analyst status are written back to the graph as provenance, so downstream consumers see human-approved intelligence. This human-in-the-loop design follows the CTI survey’s [41] central caution, that AI accelerates but cannot replace analyst judgement, and makes the AI output auditable: each claim can be traced to evidence and each report to an approver. We evaluate the summaries for evidence-reference grounding (Sect. 7).

6 Implementation

CG-CTI is deployed with Docker Compose. The knowledge base is OpenCTI 6.3.9 backed by Elasticsearch 8.15.3, Redis, RabbitMQ, and MinIO, with two ingestion workers. The seven connectors are Python modules using the OpenCTI client library and the `stix2` toolkit; each polls its source, builds a STIX 2.1 bundle, and publishes it to the platform. Conversion uses only standard STIX 2.1 types — `Bundle`, `Identity`, `Indicator`, `ObservedData`, `Sighting`, `Relationship`, `AttackPattern`, `Vulnerability`, `Malware` and the `file`, `address`, `domain`, `URL` and `network-traffic` observables — with no custom (`x_`) properties and no extension definition. Observables are wrapped in `Indicator` and `ObservedData` objects, and the standard numeric `confidence` property is carried on those (and on `AttackPattern/Vulnerability`), never on the Cyber-observable Objects themselves, which STIX 2.1 gives no `confidence`

property. The three statuses are not persisted as a STIX label: the platform stores the standard numeric **confidence** per source, and the categorical status is derived on read from the evidence the graph already records, so it is recomputed as evidence accumulates instead of being fixed at ingest. The tiers differ in how much of that vocabulary they need: detonation yields a *subject*, the artefacts it touched and the techniques it exhibited, so the sandbox connector emits the full set including **Malware**, whereas the detection and assessment connectors are **Indicator**-centred, a rule firing having no subject to attribute behaviour to. The SOC-facing read layer is a FastAPI service over the platform’s GraphQL API, and the dashboard is a React/Vite single-page application. The malware source is the CYBERGUARD live CAPEv2 (v2.4) sandbox, which the sandbox connector reaches over CAPE’s task-report API. The connector maps each report to STIX 2.1 as a **File** observable with MD5/SHA-1/SHA-256; a **Malware** object typed from the reported family; **NetworkTraffic** with **IPv4Address**, **DomainName**, and **URL** observables from the DNS/TCP/UDP/HTTP telemetry; and **AttackPattern** objects from the sandbox’s ATT&CK mapping. The detection and assessment tiers are host and network intrusion-detection sensors (Wazuh/OpenSearch and Suricata) and vulnerability and code scanners (Nmap, OWASP ZAP, Semgrep, OSV-Scanner). A separate REST wrapper around CAPE exists in the codebase but is a development scaffold and is *not* on the measured path; CG-CTI ingests directly from the sandbox’s report interface. The summarisation stage (Sect. 5) is served by a *self-hosted* Qwen2.5-14B [31] model behind an OpenAI-compatible endpoint (vLLM [23]) on operator infrastructure, so no sample, report, or generated summary leaves the organisation. This is a deliberate data-sovereignty choice for operators and national CSIRTs that cannot send malware evidence to a third-party cloud. Enrichment is the one exception: it discloses hashes and network indicators to abuse.ch, which an operator can disable at the cost of the corroboration axis.

7 Evaluation

This section evaluates CG-CTI on a labelled malware corpus detonated through the deployed CAPEv2 sandbox and ingested end-to-end into the knowledge base.

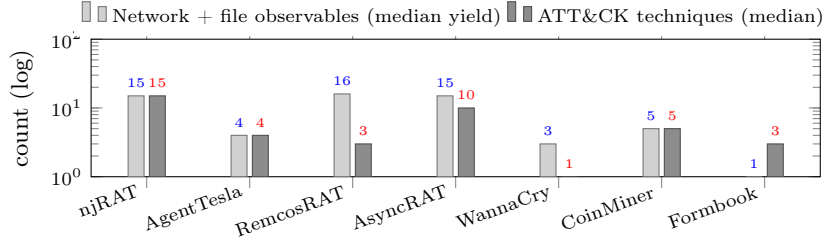
Corpus and method. We assemble a labelled corpus of 35 recent malware samples across seven families (WannaCry and CoinMiner $\times 9$, njRAT $\times 5$, AsyncRAT $\times 4$, AgentTesla and RemcosRAT $\times 3$, Formbook $\times 2$), drawn from MalwareBazaar [1] with community family labels, and detonate each through the deployed CAPEv2 sandbox under a Windows execution profile with network routing enabled. The per-family counts reflect the most recent samples available per family on MalwareBazaar at collection time; WannaCry and CoinMiner together account for 18 of the 35 and are among the lowest-yield families, which lowers the reported medians. For every sample we retrieve the sandbox report through CAPE’s task-report API and run it through the full CG-CTI pipeline into OpenCTI, so the measurements exercise the deployed system end-to-end. We answer seven questions: (Q1) *conversion validity*, what fraction of reports convert to schema-valid STIX 2.1 *objects*

— the **File**, network observables, and ATT&CK **AttackPatterns** the connector emits; the derived **Malware** object is the hub of the malware path, the source of every **drops**, **uses** and **communicates-with** edge. It anchors the graph but is not itself evidence, so it is excluded from the evaluated count: the 489 counts evidence objects, not the whole graph, which also holds the 35 **Malware** anchors and their relationships; (Q2) *yield*, how many network and file observables (contacted hosts, domains, URLs, the file hash) become shareable indicators; registry modifications are behavioural volume folded into the **Malware** object and are excluded, as are ATT&CK techniques, which become objects but are not raw observables. (Q3) *ATT&CK coverage*, how many distinct techniques are recovered from the sandbox **ttps** mapping; (Q4) *corroboration*, how often the *atomic* observables the confidence model can look up (file hash, IPs, domains) are attested by an open feed; (Q5) *confidence effect*, the confidence-status distribution over all evaluated evidence objects and the fraction eligible for automated enforcement; (Q6) *latency*, submission-to-STIX end-to-end time (p50/p95); and (Q7) *summary grounding*, what fraction of AI-summary statements are traceable to a supporting STIX object, and the rate of flagged/unsupported statements, with a light analyst check. Repository community labels are used as reference family labels; ATT&CK coverage uses the sandbox’s own technique mapping and is reported as recovered coverage, not detection accuracy.

Results. Table 2 reports the measured results. *Conversion (Q1)*: all 35 reports converted end-to-end to schema-valid STIX 2.1, 489 objects across the corpus, 100% structurally valid across the evaluated file, network, and attack-pattern objects. *Yield (Q2)*: median 5 per sample (mean 7.6, range 1–23). Registry modifications, which reach 1339 for one RAT sample (and exceed 100 for only four), are excluded as behavioural volume. Yield varies with family: interactive RATs (njRAT, AsyncRAT, RemcosRAT) surface 15–16 indicators at the median while WannaCry and Formbook surface 1–3, representative behaviour that did not fully trigger under the default profile, a coverage or execution-profile effect that is itself a signal (a low-yield sample is flagged as such, not silently under-reported) and that motivates the sandbox-hardening and multi-run directions of Sect. 10. *ATT&CK coverage (Q3)*: 39 distinct techniques are recovered across the corpus, mean 6.3 per sample. *Latency (Q6)*: end-to-end submission-to-STIX time is dominated by sandbox detonation (analysis p50=157s, p95=194s); the CAPE→STIX transform is sub-millisecond, so the intelligence is available as soon as the sandbox finishes. *Corroboration (Q4)*: 48 of 168 atomic observables (28.6%) are attested by an open feed: 34 MalwareBazaar hash hits and 14 ThreatFox *network*-indicator hits. For a MalwareBazaar-drawn corpus a file’s *presence* there is near-guaranteed (34 of 35 hashes), so hash attestation is a consistency check, not independent evidence; only the 14 ThreatFox hits on command-and-control infrastructure — not the corpus-selection key — are genuinely independent here. Community *family labels* agree with the sandbox in only 30.8% of labelled cases (an AsyncRAT tagged *XWorm*), so promotion uses feed *presence and tags*, not exact family match. *Confidence effect (Q5)*: of 489 objects, 48 (9.8%) are corroborated and hence enforce-eligible; the rest go to analyst review (behavioural

Table 2. Evaluation results on the 35-sample corpus, measured against the live CAPEv2 sandbox and the self-hosted Qwen2.5-14B LLM.

Question	Metric	Result
Q1 Conversion validity	% valid	100% (489/489 objects, 35/35 reports)
Q2 Yield (network+file)	per sample	median 5 (mean 7.6, range 1–23)
Q3 ATT&CK coverage	techniques	39 unique (mean 6.3/sample)
Q4 Corroboration	% corroborated	28.6% (48/168: 34 hash, 14 net.)
Q5 Confidence effect	enforce-eligible	48/489 objects (9.8%); rest to review/context
Q6 Latency (submit→STIX)	s (p50/p95)	157 / 194 (transform < 1 ms)
Q7 Summary grounding	grnd/flag/dropped	27.7 / 60.2 / 12.1 % (0 delivered)

**Fig. 3.** Median per-family network+file yield and technique coverage across the 35-sample corpus (log scale). Interactive RATs (njRAT, AsyncRAT, RemcosRAT) surface the most shareable indicators, whereas WannaCry and Formbook surface few under the default profile, a low-yield signal CG-CTI reports explicitly (Sect. 10).

attested) or context (uncorroborated observed). We do not claim the other 441 are false positives, only that they lack the corroboration the gate requires. Because 34 of the 48 are hash hits near-guaranteed for a MalwareBazaar-sourced corpus, the 9.8% is corpus-dependent. *Summary grounding (Q7)*: of 437 statements, 121 (27.7%) cite an object, 263 (60.2%) carry no citation and are surfaced as explicitly unsupported, and 53 (12.1%) cite a non-existent object and are *dropped* (Algorithm 1). So 0 *hallucinated references reach the analyst*, and nothing flagged is silently discarded. Both classes concentrate on the 24 lowest-yield samples; the 11 highest-yield ground 79%, so the aggregate reflects corpus skew, not model degradation.

Worked example (AgentTesla). We trace one sample end-to-end. The AgentTesla sample (SHA-256 478a9a9e..., a 32-bit Windows executable) detonates in 160s and yields 16 network and file observables (plus 18 registry modifications) and 12 ATT&CK techniques. From these the CAPE→STIX connector emits 28 STIX

Table 3. Stage-by-stage contribution (35-sample corpus): each row adds one stage and reports its measured effect. Confidence qualification narrows the enforcement set from 489 objects to 48 enforce-eligible.

Configuration	Measured output
CAPEv2 report only	raw JSON; not consumable as structured CTI
+ STIX conversion	489 schema-valid objects (100%), correlatable but undifferentiated
+ Confidence model	48/489 (9.8%) enforce-eligible; 90% routed to review or context
+ Enrichment	28.6% of atomic observables externally attested (34 MalwareBazaar consistency checks; 14 independent ThreatFox hits)
+ Grounded AI summary	12.1% fabricated references caught and dropped, 0 delivered; analyst-gated

objects, the 16 observables (a **File** carrying the sample hashes and 15 network **IPv4Address/DomainName/URL** objects for the contacted hosts and the exfiltration endpoint **ftp.freshnest2u[.]com**) together with 12 **AttackPatterns** (16+12=28) for techniques consistent with an information stealer — credentials in files (T1552.001), email collection (T1114) and application-layer command-and-control (T1071) among them; the **Malware** object anchors these relationships and, as in Q1, is not counted among the 28. The confidence stage promotes the file hash to **corroborated** once MalwareBazaar confirms the hash (recording the family tag *AgentTesla*), retains the behavioural techniques as **attested**, and marks the single-source contacted addresses **observed**. OpenCTI merges these into one graph. Finally the self-hosted LLM, given only this confidence-qualified evidence set, produces a 16-statement report of which 15 (93.8%) cite a supporting object and none reference a non-existent one, which an analyst validates before the intelligence is used. From submission to a validated, shareable product the critical path is the ~ 160 s detonation; conversion, qualification, and enrichment add sub-second overhead.

Stage-by-stage contribution. Table 3 isolates each stage’s measured contribution. The confidence model is where the enforcement set is narrowed: of 489 schema-valid objects only 48 (9.8%) are enforce-eligible, keeping 90% off the automated-action path; enrichment and the grounded summary then add corroboration and auditability. This is a stage-by-stage decomposition rather than a comparison to an external system; head-to-head baselines against raw-log LLM summarisation and converter systems (cape2stix, plain OpenCTI) remain open.

8 Related Work

CTI platforms and standards. Cross-border sharing and regulatory notification layers consume such intelligence downstream [16]. MISP and OpenCTI

are the dominant open threat-intelligence platforms, the latter natively over STIX 2.1. CG-CTI builds on OpenCTI rather than reinventing the knowledge base, contributing the malware-analysis ingestion and the confidence layer above it, and reuses STIX 2.1 and MITRE ATT&CK as interchange and vocabulary; standards-based distribution is supported by the platform but not exercised here.

Dynamic and evasive malware analysis. CAPEv2 [29], in the Cuckoo lineage, is the dynamic-analysis substrate CG-CTI consumes; the trade-offs of dynamic analysis are surveyed by Or-Meir et al. [28]. Evasion is the dominant limiting factor and an active front: hardening CAPE against VM fingerprinting [4], symbolic-execution hybrids that recover path-dependent behaviour [39], extension to Linux and OT architectures [36], and anti-VM YARA libraries [15]. CG-CTI transforms *observed* evidence, so these works bound the intelligence it can produce and motivate its flagging of low-yield samples (Sects. 7, 10).

ML-based malware representation and classification. A line of work turns low-level malware evidence into machine-learning representations: binary visualisation with a convolutional network [33] and multi-label behavioural classification over CAPE features [37]. CG-CTI is complementary: it does not classify malware but converts and operationalises its evidence into shareable intelligence.

Malware evidence to structured CTI. Closest to CG-CTI’s conversion step, cape2stix [12] deterministically maps CAPE reports to STIX 2.1 (loaded into a Neo4j graph), and AutoMAEC [42] standardises dynamic traces into malware-action semantics for power-grid control systems. CG-CTI does not re-implement a converter, its connector maps CAPE→STIX 2.1 directly, with coverage comparable to cape2stix (file, network, attack-pattern), but embeds conversion in an *operational* OpenCTI pipeline that adds confidence qualification, enrichment, graph correlation, grounded summarisation, and analyst-controlled sharing. Deterministic CAPE→STIX conversion alone is therefore no longer the novelty; the contribution is the surrounding operational lifecycle.

Confidence and quality in CTI. Confidence is not a new concern. MISP records *sightings* and decays indicators over time [40,13], and STIX 2.1 carries an optional numeric *confidence* property with standard scales (Admiralty, ICD-203) [26]; Chen et al. [7] score IOC quality across relevance, accuracy, timeliness, completeness, and source confidence; EVACTI [9] evaluates CTI *actionability*; automatic quality assessment of structured STIX separates content quality from provider confidence [38]; and weighted quality criteria *prioritise* intelligence in the MISP model [6]. These *assess* or *rank* intelligence; none makes the assessment the authority that decides whether an object may enter an automated-action path. Empirically, feeds are of uneven and often low quality [5], which motivates our lower-bound reading of feed corroboration. The same concern recurs in runtime monitoring, where an evidence-completeness status stops a gap in the event stream reading as an unqualified all-clear [19], and hierarchical edge-IoT monitors qualify verdicts similarly [20,17]; CG-CTI makes the analogous move for CTI

Table 4. Where CG-CTI sits relative to representative systems. “score” denotes an analytic quality value an analyst reads; “sightings” is MISP’s sighting count (MISP additionally *decays* indicators, akin to our durability axis); “status” a value that gates automated action. Only CG-CTI in this comparison couples per-object confidence qualification directly to an automated-enforcement gate.

System	Live sandbox	Det. STIX 2.1	Per-obj. confidence	Enforce gate	LLM stage
cape2stix [12]	–	✓	–	–	–
AutoMAEC [42]	–	MAEC	–	–	–
Chen et al. [7]	–	✓	score	–	–
EVACTI [9]	–	✓	score	–	–
MISP [40,13]	–	partial	sightings	–	–
OpenCTI [11]	–	✓	score	–	–
eLLM-CTI [30]	–	LLM	–	–	report→STIX
0-CTI [34]	–	LLM	–	–	report→STIX
CG-CTI	✓	✓	status	✓	STIX→narrative

objects. CG-CTI differs in one respect: it composes provenance, corroboration and durability into a status that *gates automated enforcement*, where prior systems record or score confidence for an analyst to read. Its axes are a minimal instance of those quality dimensions, and its evaluation measures STIX *validity*.

AI-assisted CTI and its risks. LLMs are increasingly applied to CTI: eLLM-CTI [30] and 0-CTI [34] extract STIX from unstructured reports, ChronoCTI [32] mines temporal attack patterns, and AC_MAPPER [3] classifies content into ATT&CK techniques. Recent evaluations find LLMs inconsistent and overconfident on realistic CTI [25,24], echoing the broader hallucination problem [14]. Complementary defences constrain what a model may *do* rather than say: closed intent vocabularies with policy rails [22], and runtime monitors over agent tool-call traces [18]. CG-CTI differs on both input and safety: its LLM consumes deterministically-converted, confidence-qualified STIX evidence rather than natural-language reports, and every statement is either grounded in cited evidence or marked unsupported, then analyst-validated. This restriction on the model’s role is part of the contribution.

Research gap. The gap is not that CTI lacks confidence, nor that sandbox output cannot be converted: both are well served. Prior work addresses sandbox-to-STIX conversion, CTI quality and actionability assessment, and LLM-assisted intelligence production *separately*. What is less explored, and what CG-CTI provides, is the end-to-end path: live sandbox evidence converted to structured CTI, qualified at *object* level from explicit evidence, where that qualification itself decides whether the object may enter an automated defensive-action path. The language model sits downstream of both the evidence and the control decision: it writes the analyst-facing narrative and determines neither (Table 4).

9 Discussion

Deterministic conversion, bounded AI. CG-CTI separates two concerns that report-to-STIX LLM extractors [30,34] combine: the STIX *representation* is produced deterministically, and the LLM only *summarises* evidence already converted and confidence-qualified. The model cannot invent the representation, only interpret validated objects, which is why no hallucinated reference survives the grounding check (Algorithm 1) though the model emitted them on 12.1% of statements. Given that LLMs are unreliable on realistic CTI [25,24], this bounded role and the analyst gate are what make the AI stage operationally acceptable; where a model drives action rather than narrating evidence, the same discipline moves to the action interface [22].

Confidence as a minimal quality model. The three axes reuse quality dimensions studied by Chen et al. [7] and EVACTI [9]; the contribution is not the axes but their composition into a total function that gates enforcement using only signals the graph already records. A rule-based model suits early deployment: no training labels, auditable to a SOC, and computable on data already present. Learned calibration of the cut points is a natural extension once operator feedback exists (Sect. 10).

Coverage is the ceiling. The 1–23 yield spread (Fig. 3) shows that sandbox coverage, not conversion, bounds the intelligence CG-CTI produces: on low-yield samples the pipeline surfaces unsupported content as flagged and filters fabricated references before analyst review, so invented behaviour is never presented as fact. That is the failure mode which keeps LLM output out of SOC. The same sparsity surfaces upstream as a coverage signal CG-CTI reports rather than suppresses. Evasion-resistant sandboxing [4,39], multi-architecture analysis [36], and static anti-VM detection [15] would raise the floor on yield.

Data sovereignty. Running the summarisation model on operator infrastructure (Qwen2.5-14B [31] via vLLM [23]) keeps malware evidence on-premises. For an operator that cannot send samples or derived evidence to a third-party cloud, this is a precondition for using an LLM at all.

10 Limitations and Threats to Validity

Deployment scope. CG-CTI is a project prototype evaluated against a single live CAPEv2 deployment, not yet an operational installation inside a national-CSIRT SOC, so absolute figures are indicative and the confidence and grounding behaviour await validation in production SOC/CSIRT use. *Corpus.* The evaluation corpus is detonated through the live sandbox; family labels come from the source repository and the ATT&CK figures are *recovered coverage* from the sandbox’s own technique mapping, not an independent detection-accuracy measurement. *Untuned thresholds.* The confidence cut points of Sect. 4 are design defaults. *Read*

layer. The dashboard service is pilot-grade: it currently reads indicators by string-parsing STIX patterns and labels rather than querying structured observables, and lacks authentication. *Sanitisation.* The assessment tier normalises findings but does not *sanitise* them: web-scanner findings carry internal URLs and code findings source fragments, so such objects need redaction and a marking policy before leaving the operator. *Enrichment scope.* Enrichment is limited to two open-feed lookups (passive DNS, network-ownership, geolocation and reputation sources are not consulted). *AI summaries.* Q7 measures citation *validity* for grounded statements, whether cited evidence identifiers resolve to real objects, not semantic faithfulness; the light analyst check found no unsupported claims among the grounded statements but is not a full inter-rater study. The 12.1% dropped-reference rate shows the check removes fabricated citations. *Evasion.* Dynamic analysis is defeated by anti-VM and anti-analysis malware and misses dormant code paths; the confidence model mitigates this by preferring corroborated and durable evidence but cannot recover behaviour a sample never exhibited. *Single detonation.* Each sample is detonated once under one execution profile, and yield is network- and time-dependent. *Feed trust.* The $k \geq 1$ auto-enforce gate trusts open feeds; a poisoned entry or a benign indicator listed in ThreatFox could promote a benign observable, cross-sensor corroboration is more robust than a single feed hit. *Enforcement.* CG-CTI represents the enforcement decision as an object status; wiring it to an executed control (a blocklist push to detection and response tooling) is unevaluated, so the loop is demonstrated up to the gate, not through to an executed action. *Ethics and availability.* All malware was detonated in an isolated, CYBERGUARD-controlled CAPEv2 sandbox. Network routing was enabled so samples exhibited realistic behaviour; egress was sandbox-confined with no lateral access to project or production systems, and the only external contacts were the samples' own command-and-control callbacks (reflected in the ThreatFox corroboration), not traffic directed at third parties or victims. The connectors, confidence model, and evaluation scripts will be released for replication.

11 Conclusion

CG-CTI turns the detailed but fragmented output of a malware sandbox into actionable intelligence for critical-infrastructure SOC's and national CSIRT's. It converts live CAPEv2 analyses to STIX 2.1 and correlates them in an OpenCTI knowledge graph; it attaches an explicit, evidence-derived confidence status to every object from provenance, cross-source corroboration, and observation durability, strengthened by lightweight open-feed corroboration; and it generates an evidence-grounded malware summary in which each AI statement is either traceable to the STIX evidence that supports it or explicitly marked unsupported, while fabricated evidence references are removed before analyst validation. Implemented within the CYBERGUARD project and evaluated on a labelled malware corpus detonated through a live CAPEv2 sandbox, CG-CTI shows that coupling machine-generated sandbox evidence to explicit confidence and auditable AI

turns an undifferentiated indicator stream into corroborated, durability-ranked, and explainable intelligence, an operational layer between malware analysis and actionable critical-infrastructure defence.

Future work. Four directions follow from Sect. 10. *External validity:* replicate across multiple CAPEv2 deployments and detonation profiles, separating pipeline behaviour from one sandbox’s characteristics. *Closing the enforcement loop:* execute and measure the response actions the gate authorises, and tune the default thresholds against operator feedback. *Corroboration breadth and feed trust:* widen enrichment beyond two open feeds while modelling the trust the gate places in them, since an eligibility decision inherits its sources’ weaknesses. *Semantic grounding:* extend the citation-validity check to a faithfulness study against analyst judgement, including a head-to-head against raw-log summarisation. Behavioural abstraction, multi-sample campaign narratives, and a production-grade analyst interface remain open.

Acknowledgments. This work has received funding from the European Union’s Digital Europe Programme under grant agreement No 101190251 (CYBERGUARD).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. abuse.ch: MalwareBazaar and ThreatFox: community malware and IOC feeds. <https://abuse.ch> (2024)
2. Alahmadi, B.A., Axon, L., Martinovic, I.: 99% false positives: A qualitative study of SOC analysts’ perspectives on security alarms. In: USENIX Security (2022)
3. Albarak, M., Alqudhaibi, A., Jagtap, S.: AC_MAPPER: A robust approach to ATT&CK technique classification using input augmentation and class rebalancing. *Int. J. Inf. Secur.* (2025)
4. Bayón-Martínez, R., García-Franco, D., Campazas-Vega, A., Álvarez-Aparicio, C., Guerrero-Higueras, Á.M., Matellán-Olivera, V.: Enhancing evasive malware detection through undetectable CAPE-based sandboxes. SSRN preprint 5345862 (2025)
5. Bouwman, X., Griffioen, H., Egbers, J., Doerr, C., Klievink, B., van Eeten, M.: A different cup of TI? the added value of commercial threat intelligence. In: USENIX Security (2020)
6. Chatziamanetoglou, D., Rantos, K.: Weighted quality criteria for cyber threat intelligence: assessment and prioritisation in the MISP data model. *Int. J. Inf. Secur.* (2025). <https://doi.org/10.1007/s10207-025-01080-6>
7. Chen, S.S., Hwang, R.H., Ali, A., Lin, Y.D., Wei, Y.C., Pai, T.W.: Improving quality of indicators of compromise using STIX graphs. *Comput. Secur.* (2024)
8. Dahiya, A., Singh, S., Shrivastava, G.: Android malware analysis and detection: A systematic review. *Expert Syst.* (2023). <https://doi.org/10.1111/exsy.13488>
9. Dimitriadis, A., Papoutsis, A., Kavalieros, D., Tsikrika, T., Vrochidis, S., Kompatsiaris, I.: EVACTI: Evaluating the actionability of cyber threat intelligence. *Int. J. Inf. Secur.* (2025)

10. European Parliament and Council: Directive (EU) 2022/2555 (NIS2). Official Journal of the European Union (2022)
11. Filigran: OpenCTI — open cyber threat intelligence platform. <https://github.com/OpenCTI-Platform/opencti> (2024)
12. Idaho National Laboratory: cape2stix: Automated conversion of CAPE reports to STIX 2.1. <https://github.com/idaholab/cape2stix> (2023)
13. Iklody, A., Wagener, G., Dulaunoy, A., Mokaddem, S., Wagner, C.: Decaying indicators of compromise. *arXiv:1803.11052* (2018)
14. Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y.J., Madotto, A., Fung, P.: Survey of hallucination in natural language generation. *ACM Comput. Surv.* **55**(12) (2023)
15. Kanj, S., Vila, G., Pegueroles, J.: Automating the detection of evasive Windows malware: An evaluated YARA rule library for anti-VM and anti-sandbox techniques. *J. Cybersecur. Priv.* (2026)
16. Kekatos, N., Koutidis, G., Bratsas, C., Bucur, S., Chirlejan, A.M., Curca, M., Fountoulakis, A., Ioannou, M., Lacatus, P., Lekidis, A., Charalambous, E., Michael, A., Nianios, T., Purcaru, S., Voulgaridis, A.: The CYBER-BRIDGE project: An AI-powered platform for cross-border cybersecurity, compliance and forensics in the EU. In: *IEEE CSR. IEEE* (2026)
17. Kekatos, N., Chintri, M., Katsaros, P., Lekidis, A., Nianios, T., Seitoglou, I., Basagiannis, S.: Hierarchical security monitoring for edge-IoT: A formal methods approach. In: *IEEE CSR. IEEE* (2026)
18. Kekatos, N., Chintri, M., Katsaros, P., Lekidis, A., Nianios, T., Seitoglou, I., Temperekidis, A., Basagiannis, S.: Formal runtime verification for tool-using LLM agents: An offline same-benchmark study on AgentDojo and STAC. In: *Joint Workshop on CPS & IoT Security and Privacy (CPSIoTSec). ACM* (2026), to appear
19. Kekatos, N., Chintri, M., Katsaros, P., Lekidis, A., Nianios, T., Seitoglou, I., Temperekidis, A., Basagiannis, S.: A resilient runtime-verification fabric for security monitoring of critical edge-IoT infrastructure. In: *Critical Information Infrastructures Security (CRITIS). LNCS, Springer* (2026), to appear
20. Kekatos, N., Chintri, M., Katsaros, P., Lekidis, A., Nianios, T., Seitoglou, I., Temperekidis, A., Basagiannis, S.: Hybrid hierarchical runtime verification for edge-IoT security: Combining MonPoly and RTLola. In: *Availability, Reliability and Security. ARES 2026 EU Projects Symposium Workshops*. pp. 222–240. Springer Nature Switzerland, Cham (2027). https://doi.org/10.1007/978-3-032-37218-5_14
21. Kekatos, N., Koutidis, G., Antonakopoulos, K., Basagiannis, S., Katsikas, S., Kavalieratos, G., Lekidis, A., Nianios, T., Papageorgiou, E.: RESILAGENT: A three-pillar architecture for cyber-physical resilience in electrical power and energy systems. In: *IEEE CSR. IEEE* (2026)
22. Koutidis, G., Kekatos, N., Nianios, T., Lekidis, A.: Constrained-action AI remediation for SIEM/XDR via a NeMo-Guardrails proxy. In: *IEEE CSR. IEEE* (2026)
23. Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C.H., Gonzalez, J.E., Zhang, H., Stoica, I.: Efficient memory management for large language model serving with PagedAttention. In: *ACM SOSP* (2023)
24. Meng, Y., Tang, L., Yu, F., Jia, J., Yan, G., Yang, P., Xi, Z.: Uncovering vulnerabilities of LLM-assisted cyber threat intelligence. *arXiv:2509.23573* (2025)
25. Mezzi, E., Massacci, F., Tuma, K.: Large language models are unreliable for cyber threat intelligence. In: *Availability, Reliability and Security (ARES)*. pp. 343–364. *Lecture Notes in Computer Science, Springer* (2025)

26. OASIS Cyber Threat Intelligence TC: STIX version 2.1. Oasis standard, OASIS (2021)
27. OASIS Cyber Threat Intelligence TC: TAXII version 2.1. Oasis standard, OASIS (2021)
28. Or-Meir, O., Nissim, N., Elovici, Y., Rokach, L.: Dynamic malware analysis in the modern era—a state of the art survey. *ACM Comput. Surv.* **52**(5) (2019)
29. O'Reilly, K., CAPE contributors: CAPEv2: Malware configuration and payload extraction. <https://github.com/kevoreilly/CAPEv2> (2024)
30. Psarrou, K., Bountakas, P., Eleutheriou, D., Ellinitakis, R.A., Fysarakis, K., Lekidis, A., Spanoudakis, G.: Enhanced-LLM extraction of CTI from unstructured threat reports. a tough nut to crack or a walk in the park? *Future Gener. Comput. Syst.* **183**, 108514 (2026)
31. Qwen Team: Qwen2.5 technical report. *arXiv:2412.15115* (2024)
32. Rahman, M.R., Wroblewski, B., Matthews, Q., Morgan, B., Menzies, T., Williams, L.: Mining temporal attack patterns from cyberthreat intelligence reports. *Knowl. Inf. Syst.* **67**, 8941–8981 (2025)
33. Shire, R., Shiaeles, S., Bendiab, K., Ghita, B., Kolokotronis, N.: Malware squid: A novel IoT malware traffic analysis framework using convolutional neural network and binary visualisation. In: *NEW2AN/ruSMART* (2019)
34. Sorokoletova, O., Antonioni, E., Colò, G.: Towards a scalable AI-driven framework for data-independent cyber threat intelligence information extraction. In: *IEEE Int. Conf. on Foundation and Large Language Models (FLLM)*. pp. 398–406 (2024)
35. Strom, B.E., Applebaum, A., Miller, D.P., Nickels, K.C., Pennington, A.G., Thomas, C.B.: MITRE ATT&CK: Design and philosophy. Tech. Rep. MP180360R1, The MITRE Corporation (2020)
36. Udeshi, M., Putrevu, V.S.C., Krishnamurthy, P., Karri, R., Khorrami, F.: SaMOSA: Sandbox for malware orchestration and side-channel analysis. *arXiv:2508.14261* (2025)
37. Uysal, D.T., Yoo, P.D., Taha, K., Yeun, C.Y., Damiani, E.: A multi-label visualisation approach for malware behaviour analysis. *Sci. Rep.* **15** (2025)
38. Venčkauskas, A., Jusas, V., Barisas, D.: Quality dimensions for automatic assessment of structured cyber threat intelligence data. *Appl. Sci.* **15**(8), 4327 (2025). <https://doi.org/10.3390/app15084327>
39. Vouvoutsis, V., Casino, F., Patsakis, C.: Beyond the sandbox: Leveraging symbolic execution for evasive malware classification. *Comput. Secur.* (2025)
40. Wagner, C., Dulaunoy, A., Wagener, G., Iklody, A.: MISP: The design and implementation of a collaborative threat intelligence sharing platform. In: *ACM WISCS* (2016)
41. Wagner, T.D., Mahbub, K., Palomar, E., Abdallah, A.E.: Cyber threat intelligence sharing: Survey and research directions. *Comput. Secur.* **87** (2019)
42. Yu, T., Wu, H., Li, W., Lian, Y.: AutoMAEC: An automated framework for semantic-aware malware characterization in power grid industrial control systems. *J. Eng. Appl. Sci.* (2026)