# From Attack Scenario to Measurable Detection Improvement: A Knowledge-Driven ATT&CK / D3FEND Framework for Instrumented Purple Teaming

Tristan Madani[2], Yulliwas Ameur[1,2] and Samia Bouzefrane[2]

[1]*Efrei Research Lab, Efrei, Université Paris-Panthéon-Assas, Villejuif, France*

[2]*Centre détudes et de recherche en informatique et communications (Cédric), Conservatoire national des arts et métiers (Cnam), Paris, France*

**Abstract**

Organizations run purple team exercises, find gaps, patch a few rules, and report success—but without a formal measurement framework, they cannot quantify *how much* detection improved or *why* specific techniques went undetected. Attack prediction models achieve strong F1 scores but produce no defensive output, while CTI knowledge graphs map threats to countermeasures without testing them on real telemetry.

The gap between *emulating an attack* and *proving that detection improved* remains open.

We close this gap with a knowledge-graph-driven framework for instrumented purple teaming. A *bidirectional knowledge graph* encodes the full chain from ATT&CK technique to data source, detection strategy, analytic rule, log source, and sensor—not just the attack side. Each emulation step is annotated with expected telemetry and success criteria, so that when detection fails, backward traversal of the graph pinpoints *why*: a missing log source, an absent rule, or an inadequate configuration. After targeted remediation, the same scenario is replayed and improvement is quantified through operational KPIs anchored by detection coverage ($C$).

We evaluate on four threat profiles—nation-state espionage (APT29), financial cybercrime (FIN6), state destructive operations (Sandworm), and big-game-hunting ransomware (Wizard Spider)—all sourced from the public CTID adversary emulation library [1]. The primary endpoint is detection coverage ($C$), evaluated with a fixed-threshold decision rule ($\Delta C > 0.10$ in $\geq$3/4 scenarios). The four-step detection maturity ladder raises strict coverage from $C_0 = 23.1\%$ (out-of-the-box SIEM) to $C_3 = 71.6\%$ (with KGPT (Knowledge-Graph Purple Teaming)-guided rules), with the KG-guided rule engineering step alone contributing $\Delta C_3 = +30.3$ percentage points on average. The decision rule is satisfied in all four scenarios.

**Keywords**

adversary emulation, ATT&CK, D3FEND, knowledge graph, detection engineering, purple teaming, detection coverage

## 1. Introduction

Security teams run purple team exercises, find gaps, fix some rules, and move on. But they cannot answer a basic question: *did detection actually improve, and by how much?* Despite mature ATT&CK/D3FEND taxonomies and widely used emulation and rule formats, organizations still lack a closed-loop framework that links emulation to measurable detection improvement.

Even with ATT&CK and MITRE Engenuity as common benchmarks, organizations still lack a systematic way to measure their *own* detection posture against specific scenarios.

Data from 160+ purple team exercises confirms systematic gaps between what organizations *think* they detect and what they *actually* detect [2].

Three research streams address parts of this problem, but each stops short:

- **Attack prediction** (e.g., DeepOP [3], F1 = 0.894): models adversary sequences but produces no defensive output—a predicted technique is not a detected one.

*C&ESAR 2026: Computer & Electronics Security Application Rendezvous, Rennes, France*

tristan.madani@lecnam.net (T. Madani); yulliwas.ameur@efrei.fr (Y. Ameur); samia.bouzefrane@lecnam.net (S. Bouzefrane)

0009-0004-1476-9475 (T. Madani); 0000-0003-1435-2982 (Y. Ameur); 0000-0002-0979-1289 (S. Bouzefrane)

- **CTI knowledge structuring** (e.g., K-CTIAA [4], F1 = 0.941): maps threat actions to ATT&CK/D3FEND countermeasures, but never tests whether those countermeasures fire on real telemetry.
- **Purple teaming**: improves detection in practice through "execute–observe–correct–replay" cycles [2], but with ad-hoc metrics that cannot be compared across exercises or organizations.

The missing piece is a knowledge structure that encodes not just *what* to attack, but *what should detect it, how, and where the detection chain can break*. When a detection fails, the analyst needs to traverse backward from the missed technique to the specific gap: was the log source unconfigured? Was there no rule? Was the rule present but too narrow? Today, this diagnosis is manual and expertise-dependent.

We propose a framework that makes it systematic:

- A **bidirectional knowledge graph** (Section 3) encoding the full chain from ATT&CK technique through data component, detection strategy, analytic rule, log source, to sensor. When detection fails, backward graph traversal pinpoints the break.
- **Instrumented emulation scenarios** (Section 4) spanning four threat profiles—nation-state espionage (APT29), cybercrime (FIN6), state destructive operations (Sandworm), and big-game-hunting ransomware (Wizard Spider)—each annotated with expected telemetry, target rules, and measurement hooks.
- A **within-subject purple teaming protocol** (Section 5) with a reproducible testbed, a four-step detection maturity ladder, and a fixed-threshold decision rule. The Phase 3 rule-engineering step is constrained by a frozen rule-authoring protocol that requires every custom rule to derive from ATT&CK sub-technique documentation.
- **Operational KPIs** (Section 6) anchored by Detection Coverage ($C$) for rigorous before/after comparison across detection layers.

## 2. Related Work

### 2.1. Attack Modeling and Sequence Prediction

DeepOP [3] proposes the COASE ontology with causal window attention, achieving F1 = 0.894 on next-technique prediction; however, the framework operates entirely on the attack side with no connection to detection. Other approaches—attack graphs [5], attack-defense trees [6], HMMs, and GNNs—share this limitation of prediction-only outputs [7].

### 2.2. CTI Extraction and Knowledge Graphs

K-CTIAA [4] augments BERT with cybersecurity KGs (F1 = 0.941 on threat action extraction) and maps extracted actions to D3FEND countermeasures, but the countermeasure module is never quantitatively evaluated and no connection to operational telemetry exists. Broader unified KG efforts [8, 9, 10, 11] integrate ATT&CK, D3FEND, ENGAGE, CWE, and CVE but none include operational elements such as log sources, sensor configurations, or detection analytics.

### 2.3. Detection Coverage and Purple Teaming

MITRE Engenuity ATT&CK evaluations [12] benchmark EDR products in controlled conditions but do not measure an organization's own detection posture. Winkler and Sharma [13] measure Wazuh detection against ATT&CK-emulated attacks (85% detection rate)—the closest existing work to our before/after approach, though without an iterative improvement cycle. DeTT&CT [14] scores visibility against ATT&CK but operates as a manual framework without integrated emulation.

The closest methodological prior art is the two-pass Detection/Protection structure in the CTID Sandworm and Wizard Spider emulation plans [1], which applies the same within-subject baseline-vs-hardened logic that underlies our protocol. Our contributions relative to these plans are: (i) a

bidirectional KG that automates per-technique failure-mode diagnosis, (ii) a pre-registered decision rule instead of descriptive walkthroughs, (iii) cross-actor evaluation spanning four motivational profiles, and (iv) a frozen rule-authoring protocol bounding operator bias.

### 2.4. Adversary Emulation Platforms

CALDERA [15] is the reference open-source emulation platform; Atomic Red Team [16] provides unit-test-style procedures for individual techniques; MITRE Engenuity's adversary emulation plans [1] offer gold-standard playbooks. Laccolith [17] shows that emulation tools can be detected by AV, highlighting the need for emulation realism.

### 2.5. Detection Engineering

The detection-as-code movement, anchored by Sigma [18], treats rules as versionable software artifacts. Recent work explores LLM-assisted rule optimization [19] and benchmarks for rule generation [20]. D3FEND validation remains sparse [21]; no work validates D3FEND mappings against empirical detection data.

### 2.6. Gap Statement

No existing work unifies KG-driven scenario derivation, instrumented emulation, a within-subject protocol with statistical rigor, and quantitative before/after measurement. Table 1 summarizes the positioning.

**Table 1**
Positioning of our framework relative to existing work.

| Dimension | DeepOP | K-CTIAA | CTID Sandworm | Purple Team | Ours |
|---|---|---|---|---|---|
| Attack modeling | ✓✓ | ✓ | ✓ | ✓ | ✓✓ |
| Knowledge graph | Attack-only | +D3FEND | — | — | Bidirectional |
| Telemetry / detection | — | — | ✓ | ✓ | ✓✓ |
| Operational KPIs | Prediction F1 | Extraction F1 | Walkthroughs | Informal | 6 defined |
| Emulation | — | — | Two-pass | Ad-hoc | Instrumented |
| Before/after loop | — | — | Manual | Informal | Protocol-frozen |
| KG-traceable diagnosis | — | — | — | — | ✓✓ |
| Cross-actor evaluation | 1 dataset | 1 dataset | 1 actor | varies | 4 actors |

## 3. Knowledge Graph Construction

### 3.1. Schema

Existing cybersecurity ontologies model either the attack side (COASE [3]) or provide static defensive mappings (K-CTIAA [4]). The architectural insight of this work is that a single property graph can encode *both sides plus the operational telemetry chain*—data component, log source, sensor—so that a failed detection can be explained in one backward Cypher traversal. We adopt a property graph model (Neo4j) with 14 entity types and 14 relation types.

#### 3.1.1. Entity Types.

Table 2 defines the core entities, grouped by domain.

**Table 2**
Knowledge graph entity types.

| Domain | Entity | Key Properties |
|---|---|---|
| Threat | Tactic | tactic_id, name, kill_chain_phase |
| | ThreatActor | name, aliases, motivation |
| | Campaign | name, date_range, target_sectors |
| | Technique | technique_id, name, platforms, detection_desc |
| | Procedure | description, tool_used, sequence_position |
| Defense | DataSource | name, platform |
| | DataComponent | name, parent_data_source |
| | DetectionStrategy | strategy_id, analytic_type |
| | DetectionAnalytic | rule_id, title, format (Sigma/SPL/KQL), FP rate |
| D3FEND | D3FENDTechnique | d3fend_id, name, d3fend_tactic |
| | DigitalArtifact | artifact_id, name, type |
| Infra | LogSource | source_type, event_ids |
| | Sensor | sensor_name, coverage_scope |
| Emulation | EmulationTest | test_id, executor, platform, cleanup |

### 3.1.2. Relation Types.

The defensive backbone encoding "what does it take to detect this technique?" uses four edges with directions matching the ingestion schema:

$$\texttt{Technique} \xrightarrow{\text{GENERATES\_ARTIFACT}} \texttt{DataComponent} \xleftarrow{\text{REQUIRES\_DATA}} \texttt{DetectionAnalytic}$$
$$\texttt{DataComponent} \xrightarrow{\text{COLLECTED\_BY}} \texttt{LogSource} \xrightarrow{\text{MONITORED\_BY}} \texttt{Sensor}$$

A single forward traversal answers: "For technique $T$, which data components are produced, which log sources collect them, which sensors are needed, and which analytics consume the result?" A single backward traversal from a missed technique answers *why* it was missed (no log source, no analytic, or the analytic's logic is too narrow).

The remaining ten relation types connect the threat domain (BELONGS_TO_TACTIC, USES_TECHNIQUE, PART_OF_CAMPAIGN), the detection domain (DETECTED_BY_STRATEGY, IMPLEMENTED_BY), the D3FEND bridge (COUNTERED_BY, OPERATES_ON), emulation (EMULATED_BY), and sequencing (SEQUENCE_PRECEDES auto-derived, CAUSALLY_PRECEDES hand-authored, ∼32 cross-tactic prerequisite edges).

## 3.2. Population Method

The KG is populated in three phases:

**Phase A: Automated ingestion.** ATT&CK STIX bundles (v16) are parsed with `mitreattack-python`; D3FEND OWL ontology with `rdflib`; Sigma rules (SigmaHQ master) and Atomic Red Team YAML definitions with `pyyaml`. All input versions are pinned by commit SHA in the repository. This automated phase populates the majority of nodes and edges.

**Phase B: Campaign enrichment.** For each actor, technique sequences are imported from the CTID adversary emulation library [1]. Sequential ordering is auto-derived from sub-step numbering; ∼32 hand-authored CAUSALLY_PRECEDES edges capture genuine cross-tactic prerequisites.

**Phase C: Gap analysis.** Completeness queries identify *blind spots* (techniques with no detection analytic) and *dark telemetry* (data components with no configured log source in the testbed). These gaps directly inform the detection engineering cycle in Section 5.

Scoped to the four primary scenarios, the populated graph contains 201 ATT&CK technique nodes (v16.0), 121 procedure nodes derived from the four CTID emulation plans, 3,785 detection analytic nodes (1,389 Elastic prebuilt + 2,396 SigmaHQ), 43 data-component nodes matching the ATT&CK data source taxonomy, and 154 D3FEND technique nodes, yielding approximately 4,800 nodes and 19,000 edges. The Phase C gap analysis identifies 93 of the 121 scenario-specific procedures (76.9%) as lacking a correctly-mapped detection analytic at the $C_0$ baseline—consistent with the empirically measured $C_0 = 23.1\%$ strict coverage reported in Section 7.1.

# 4. Instrumented Emulation Scenarios

## 4.1. Scenario Selection

We select four scenarios spanning distinct profiles of operational intent to test consistency of the methodology across diverse threat models. All plans are sourced from the CTID adversary emulation library [1] (Apache-2.0).

1. **S1 — APT29** (nation-state espionage). Stealthy, living-off-the-land techniques across a broad kill chain (≈79 procedures, 14 tactics).
2. **S2 — FIN6** (financial cybercrime). Commodity tooling, fast lateral movement, credential theft and POS data staging (27 procedures, 9 tactics).
3. **S3 — Sandworm** (state destructive operations). NotPetya-style wiper with lateral propagation; tests detection *before* destructive action (≈40 procedures, 11 tactics).
4. **S4 — Wizard Spider** (big-game-hunting ransomware). Multi-stage `Emotet`→`TrickBot`→`Ryuk` chain terminating in bulk encryption; tests identification of encryption precursors (38 procedures, 9 tactics).

The diversity we claim is *motivational* (espionage, cybercrime, destruction, extortion), not geographic—three of the four actors are attributed to Russian state or criminal ecosystems, a consequence of CTID's current plan availability (see Threats to Validity, §8.4).

## 4.2. Instrumentation Records

Each technique in a scenario is annotated with an *Instrumentation Record (IR)* specifying:

- **Offensive side:** emulation command (ART/CALDERA), executor, input arguments, cleanup procedure.
- **Defensive side:** expected data components and log sources, target detection analytics (Sigma rules), D3FEND countermeasures.
- **Measurement hooks:** timestamp markers ($T_{\text{exec}}$, $T_{\text{log}}$, $T_{\text{alert}}$), success criteria for telemetry presence and alert quality.

The IR schema is derived automatically from KG traversal: given a technique node, the graph provides all reachable defensive entities and emulation tests.

Table 3 summarizes the per-scenario statistics as computed from the CTID plans (`master` branch, 2026-04-14).

A worked IR example is provided in Appendix A. Pre-execution Sigma coverage matrices for all four scenarios are computed by `gap_analysis.py`.

# 5. Purple Teaming Protocol

## 5.1. Testbed Architecture

The testbed is a fully isolated lab of seven VMs (1 DC, 1 member server, 2 Windows 10 workstations, 1 Linux application server, 1 SIEM, 1 C2/orchestrator) running on a single physical host under

**Table 3**
Per-scenario statistics for the four primary emulation plans. Procedure counts are the number of *executable* steps in the CTID YAML (steps marked as requiring user interaction or external C2 are excluded). Tactic and technique counts are the number of unique MITRE ATT&CK tactics/techniques covered by at least one executable procedure.

| Scenario | Procs. | Tactics | Techniques | Profile |
|---|---|---|---|---|
| S1 — APT29 | ≈79 | 14 | ≈60 | nation-state espionage |
| S2 — FIN6 | 27 | 9 | 15 | cybercrime (POS, exfil) |
| S3 — Sandworm | ≈40 | 11 | ≈30 | state destructive (wiper) |
| S4 — Wizard Spider | 38 | 9 | 24 | big-game-hunting ransomware |
| **Total** | **≈184** | – | – | 4 distinct profiles |

QEMU/KVM. Two network segments enforce isolation: a TARGET VLAN with no Internet egress (verified per-run via `tracepath`) and a MGMT network for operator access. Full hardware specifications, VM configurations, and provisioning runbooks are published in the artifact repository.

**Telemetry stack (fixed across all four steps).** The telemetry infrastructure is held constant across all steps; only detection layers vary. On Windows: Sysmon v15.20 [22] (full-coverage configuration: EID 1, 3, 5, 7, 8, 10, 11, 12–14, 22), Winlogbeat 8.13.0 with a 39-processor ECS ingest pipeline, PowerShell Script Block Logging (EID 4104), and Windows Security audit policies (4624/4625/4688/4698/5156). On Linux: `auditd` covering `execve`, credential files, and SUID binaries. SIEM is Elastic Security [23] (Elasticsearch 8.x + Kibana); Sigma rules are converted via pySigma [24] and loaded as Kibana detection rules.

**Execution and snapshots.** Techniques are executed via WinRM (pywinrm, NTLM transport) with three-point timestamping ($T_{\text{exec}}$, $T_{\text{log}}$, $T_{\text{alert}}$). VMs are reverted to golden snapshots between runs. A benign-noise generator produces realistic background traffic on all targets to make FPR measurement meaningful.

## 5.2. Execution Protocol

Each scenario follows a five-phase cycle: (1) *Pre-flight*—verify sensor and SIEM integrity, snapshot all VMs; (2) *Baseline run*—execute the emulation playbook with no prior detection engineering, record KPIs, restore snapshots between runs; (3) *KG-guided gap analysis*—for each missed technique, query the KG for the defensive chain, identify the failure mode (missing telemetry, missing rule, or narrow rule), and apply targeted fixes documented as KG-traceable remediation actions; (4) *Post-adjustment run*—re-execute the identical playbook with the improved configuration; (5) *Comparative analysis*—compute per-KPI deltas. Network topology diagrams, Sysmon configurations, and execution logs are published in the artifact repository (Section 10).

**Phase 3 rule-authoring protocol.** Phase 3 is unblinded by design: rule authoring must respond to observed failures. We bound operator bias procedurally: (i) every rule must derive from ATT&CK sub-technique documentation, not from observed alert fields; (ii) every rule is reviewed by an independent second rater who has not observed the baseline; (iii) every rule is rejected if it does not pass a rule-specificity gate. The full protocol is published in the artifact repository (Section 10).

## 6. Key Performance Indicators

The primary KPI is **Detection Coverage ($C$)**: the proportion of emulated techniques for which at least one detection analytic fires a correctly-mapped true-positive alert:

$$C = \frac{|\{t \in T_{\text{scenario}} : \text{detected}(t)\}|}{|T_{\text{scenario}}|}$$

To account for statistical uncertainty, detection coverage is modeled as a binomial proportion and 95% confidence intervals are computed using the Wilson score interval. This provides a robust estimate of variability given the binary nature of detection outcomes at the technique level.

Five additional KPIs—Alert Quality ($Q$), Detection Delay ($D$), Robustness ($R$), Chain Tracking Completeness ($\Gamma$), and False Positive Rate ($\Phi$)—are formally defined in the experimental protocol. While their full evaluation requires repeated runs, the primary conclusions are supported by consistent coverage improvements across all scenarios. Additionally, pairwise comparisons between detection steps (e.g., $C_2$ vs. $C_3$) show statistically significant improvements under standard two-proportion tests, confirming the robustness of the observed gains.

**Decision rule.** The pre-registered primary endpoint is $\Delta C_3 = C_3 - C_2 > 0.10$ (10 percentage points) in at least 3 of the 4 primary scenarios. Because each technique's detection is a deterministic binary outcome at a given step, $C$ is a population parameter over the scenario's technique set; the fixed-threshold decision rule evaluates this rate directly (Amendment 8).

## 7. Results

The experimental protocol follows a four-step detection maturity ladder. Each step adds one layer of detection capability while keeping telemetry and emulation playbooks identical, isolating the marginal contribution of each layer:

**Step 1** SIEM + Elastic prebuilt detection rules only $\rightarrow C_0$
**Step 2** + SigmaHQ community rules (converted via pySigma) $\rightarrow C_1$
**Step 3** + EDR (Elastic Defend, detect-only mode) $\rightarrow C_2$
**Step 4** + KGPT-generated Sigma rules from KG gap analysis $\rightarrow C_3$

**Terminology.** KGPT (Knowledge-Graph Purple Teaming) denotes a graph-guided detection engineering loop: gap diagnosis and telemetry-path selection are automated via KG traversal, while rule logic is authored under a frozen protocol.

The primary endpoint remains $\Delta C_3 = C_3 - C_2 > 0.10$ in at least 3 of the 4 primary scenarios. The ladder design allows us to attribute detection gains to each layer individually, strengthening the causal claim for the KG-guided rule engineering step.

### 7.1. Step 1 Baseline — SIEM + Elastic Prebuilt Rules ($C_0$)

Step 1 establishes the $C_0$ baseline: what an organization detects with an out-of-the-box SIEM deployment (Elasticsearch 8.x + Kibana with 1,389 Elastic prebuilt detection rules enabled) and standard telemetry (Sysmon v15.20 + Winlogbeat 8.13.0 with a custom ECS ingest pipeline). No community Sigma rules, no EDR, and no custom rules are present. Windows Defender real-time protection is disabled on all targets to allow commodity tools to execute without AV interference (this substitution is identical across all four steps and cancels in $\Delta C$, as discussed in §8.3).

Techniques were executed via WinRM (pywinrm, NTLM transport) rather than CALDERA orchestration due to agent instability. Each technique maps to exactly one command; alerts are collected after a 5-minute rule execution window.

**Detection outcome classification.** For each technique, we cross-reference the executed command line against both the raw telemetry index (`winlogbeat-*`) and the alert index, yielding four categories: **DET** (alert fired with correct ATT&CK mapping), **DET*** (alert fired but mapped to a different technique—operationally useful but a gap-analysis blind spot), **TEL** (telemetry present but no rule fired), and **NONE** (no telemetry ingested). The TEL/NONE distinction is critical for KG-guided gap analysis: TEL requires a new rule; NONE requires a new telemetry source.

Table 4 summarises the Step 1 results across all four primary scenarios, and Table 5 breaks down the detection outcomes by classification.

**Table 4**
Step 1 detection coverage ($C_0$) per scenario. "Exec" = techniques executed successfully; "Det" = techniques for which at least one correctly-mapped Elastic prebuilt rule fired.

| Scenario | Threat Profile | Techniques | Det | Missed | $C_0$ |
|---|---|---|---|---|---|
| S1-APT29 | Nation-state espionage | 28 | 6 | 22 | 21.4% |
| S2-FIN6 | Financial cybercrime | 26 | 5 | 21 | 19.2% |
| S3-Sandworm | State destructive ops | 30 | 9 | 21 | 30.0% |
| S4-WizardSpider | Ransomware (big-game) | 37 | 8 | 29 | 21.6% |
| **Average** | | **30.25** | **7** | **23.25** | **23.1%** |

**Table 5**
Step 1 detection outcome classification per scenario. Each executed technique is classified by cross-referencing its specific command line against both the raw telemetry index and the alert index.

| Scenario | DET | DET* | TEL | NONE |
|---|---|---|---|---|
| S1-APT29 | 6 (21.4%) | 1 (3.6%) | 21 (75.0%) | 0 |
| S2-FIN6 | 5 (19.2%) | 0 | 21 (80.8%) | 0 |
| S3-Sandworm | 9 (30.0%) | 0 | 21 (70.0%) | 0 |
| S4-WizardSpider | 8 (21.6%) | 0 | 29 (78.4%) | 0 |

**Telemetry completeness.** Zero techniques fall into the NONE category across all 121 executions—every technique produced at least one Sysmon, Security audit, or PowerShell event in the SIEM. The detection gap is entirely in the *rule layer*, validating the design choice of holding telemetry constant across all four steps.

**Consistently detected techniques.** Four prebuilt rules fire across all scenarios: PowerShell arguments (T1059.001), process discovery (T1057), system info discovery (T1082), and credential access (T1003.001). S3/S4 add log-wiping (T1070.001) and file deletion (T1070.004).

**Rule coverage gap.** 70–81% of techniques are TEL: the SIEM ingested relevant events but no prebuilt rule fired. This spans lateral movement, persistence, registry modification, defense evasion, and discovery—prime candidates for SigmaHQ (Step 2) and KGPT rules (Step 4).

**Cross-scenario observations.** S3-Sandworm achieves the highest $C_0$ (30.0%) because destructive operations trigger high-severity prebuilt rules; S2-FIN6 has the lowest (19.2%) due to less conspicuous lateral-movement techniques. The 10.8 pp spread motivates the multi-scenario design.

## 7.2. Step 2 — + SigmaHQ Community Rules ($C_1$)

Step 2 adds 2,396 community detection rules from the SigmaHQ repository (commit `df5c6a6`, 2026-05-12). Rules were converted from the Sigma YAML format to Lucene queries via pySigma 1.3.3 [24], using

the `ecs_windows` pipeline and the Elasticsearch backend [25] (pySigma-backend-elasticsearch 2.0.2). Converted rules were then loaded as Kibana detection rules targeting the `winlogbeat-*` index pattern, matching the existing telemetry stack. Only Windows-applicable rules in `test` or `stable` status were included; deprecated and unsupported rules were excluded. Zero conversion failures occurred. All prebuilt Elastic rules from Step 1 remain active alongside the SigmaHQ corpus.

After loading, all four scenarios were re-executed using identical WinRM commands and the same deep-audit classification methodology as Step 1. Tables 6 and 7 present the results.

**Table 6**
Step 2 detection coverage ($C_1$) per scenario. $C_1^{\text{strict}}$ counts only correctly-mapped alerts (DET); $C_1^{\text{eff}}$ includes alerts under different ATT&CK mappings (DET + DET$^*$). $\Delta$ columns show the change from $C_0$.

| Scenario | $n$ | $C_1^{\text{strict}}$ | $\Delta C_1^{\text{strict}}$ | $C_1^{\text{eff}}$ | $\Delta C_1^{\text{eff}}$ |
|---|---|---|---|---|---|
| S1-APT29 | 28 | 21.4% | $\pm$0.0 pp | 57.1% | +32.1 pp |
| S2-FIN6 | 26 | 19.2% | $\pm$0.0 pp | 53.8% | +34.6 pp |
| S3-Sandworm | 30 | 30.0% | $\pm$0.0 pp | 63.3% | +33.3 pp |
| S4-WizardSpider | 37 | 21.6% | $\pm$0.0 pp | 54.1% | +32.4 pp |
| **Average** | **30.25** | **23.1%** | **$\pm$0.0 pp** | **57.0%** | **+33.1 pp** |

**Table 7**
Step 2 detection outcome classification per scenario.

| Scenario | DET | DET$^*$ | TEL | NONE |
|---|---|---|---|---|
| S1-APT29 | 6 (21.4%) | 10 (35.7%) | 12 (42.9%) | 0 |
| S2-FIN6 | 5 (19.2%) | 9 (34.6%) | 12 (46.2%) | 0 |
| S3-Sandworm | 9 (30.0%) | 10 (33.3%) | 11 (36.7%) | 0 |
| S4-WizardSpider | 8 (21.6%) | 12 (32.4%) | 17 (45.9%) | 0 |

**Strict coverage is unchanged.** $C_1^{\text{strict}} = C_0$ in all four scenarios: 2,396 SigmaHQ rules produce *zero* new correctly-mapped detections. No community rule fires with the correct ATT&CK annotation for any previously undetected technique.

**Effective coverage approximately doubles.** 9–12 techniques per scenario transition from TEL to DET$^*$, yielding $\Delta C_1^{\text{eff}} = +33.1$ pp on average. The TEL category shrinks from 70–81% to 37–46%; NONE remains zero.

**The mapping-quality gap.** Manual inspection of the 17 unique SigmaHQ rules responsible for DET$^*$ outcomes reveals that most fire on the *execution transport* (WinRM lateral-movement rules) rather than the *technique payload*. Tool-specific rules correctly identify the tool but annotate at the parent technique level (e.g., T1053 instead of T1053.005). Detection logic is high-quality, but ATT&CK metadata is approximate—organizations relying on technique-level coverage dashboards would still report these techniques as undetected.

**Implications for Step 4.** The 11–17 techniques still in TEL need only a correctly-mapped rule; the DET$^*$ techniques are improvement targets where KGPT rules with precise annotations can promote them to DET.

### 7.3. Step 3 — + EDR Detect-Only ($C_2$)

Step 3 adds Elastic Defend 8.19 in detect-only mode (malware, ransomware, memory-protection, and behavior-protection all set to `detect`), following the EDR telemetry evaluation methodology of Virkud

et al. [26]. The agent is enrolled on all three Windows targets (DC01, WS01, WS02) via Fleet Server. Critically, blocking is disabled: every technique still executes to completion, preserving comparability with Steps 1–2. The existing Sysmon + Winlogbeat telemetry and the full rule set (1,389 prebuilt + 2,396 SigmaHQ) remain unchanged.

**Table 8**
Step 3 per-scenario coverage ($C_2$) with EDR.

| Scenario | Exec | DET | DET* | TEL | NONE | $C_2$ strict | $C_2$ eff |
|---|---|---|---|---|---|---|---|
| S1-APT29 | 28 | 9 | 12 | 7 | 0 | 32.1% | 75.0% |
| S2-FIN6 | 26 | 12 | 7 | 7 | 0 | 46.2% | 73.1% |
| S3-Sandworm | 30 | 14 | 8 | 8 | 0 | 46.7% | 73.3% |
| S4-WizardSpider | 37 | 15 | 12 | 10 | 0 | 40.5% | 73.0% |
| **Average** | **30.25** | **12.5** | **9.75** | **8** | **0** | **41.4%** | **73.6%** |

**Key observations.** The average $C_2 = 41.4\%$ represents +18.3 pp over $C_1$, driven by six unique Elastic Defend behavioral rules: LSASS dump detection (T1003.001), renamed binary proxy (T1036.005), suspicious scheduled task (T1053.005), ingress transfer (T1105), and registry run key (T1547.001). Three techniques (T1036.005, T1105, T1547.001) transition from TEL/DET* to DET. Effective coverage converges to 73–75% across all scenarios (vs. 53.8–63.3% at Step 2), producing a uniform detection floor.

7–10 techniques per scenario remain TEL: impact-phase techniques (T1486, T1489, T1490), credential techniques requiring memory analysis (T1003.003, T1555.003), and defense-evasion (T1562.001).

### 7.4. Step 4 — + KGPT-Generated Rules from KG Gap Analysis ($C_3$)

Step 4 adds custom Sigma detection rules generated through the KGPT knowledge-graph gap analysis. For each technique still in the TEL category after Step 3 (7–10 per scenario), backward traversal from the technique node through its data-component and log-source edges identifies the failure mode and the correct remediation path. This analysis produces two complementary categories of rules.

**Process-creation rules (16 rules).** The first set targets Sysmon EID 1 and Security EID 4688 events via the `process_creation` Sigma log source category, matching on `process.command_line` patterns. These cover techniques where the command-line string is the primary detection indicator: lateral tool transfer via `copy`/`xcopy` (T1570), network share enumeration via `net share`/`net view` (T1135), registry modification via `reg add` (T1112), service stop via `net stop`/`sc stop` (T1489), data destruction via file operations (T1485), inhibit system recovery via `vssadmin delete shadows` (T1490), and file rename to known ransomware extensions (T1486).

**PowerShell Script Block rules (10 rules).** The KG backward traversal from FIN6 TEL techniques revealed a systematic gap: five techniques (T1005, T1047, T1071.001, T1552.001, T1562.001) execute exclusively via WinRM `run_ps()`, which delivers PowerShell to the target as a script block rather than a new process. These commands populate PowerShell Event ID 4104 (`ScriptBlockText`) but *not* Sysmon EID 1 (`process.command_line`). Process-creation rules are structurally blind to this execution path. A second set of 10 rules was authored targeting the `ps_script` Sigma category, matching on `ScriptBlockText` content: `Get-ChildItem` with credential-file extensions (T1552.001), `Set-MpPreference -Disable*` (T1562.001), `Invoke-WmiMethod` (T1047), `Copy-Item` to temp or admin shares (T1570), and web request cmdlets (T1071.001).

This dual-category approach demonstrates the practical value of the KG backward traversal: the gap is not a missing rule but a missing *log-source path*. A practitioner authoring rules without graph-guided diagnosis would likely write only process-creation rules and miss the WinRM/ScriptBlock detection path entirely.

All 26 rules follow the frozen Phase 3 rule-authoring protocol: each derives from ATT&CK sub-technique documentation, not from observed playbook commands. Every rule cites the ATT&CK technique URL in its source header and undergoes independent second-rater review. Rules are converted to Lucene via pySigma and deployed to Kibana (30-minute lookback, 5-minute interval), targeting the `winlogbeat-*` and `logs-windows.powershell*` index patterns.

Tables 9 and 10 present the Step 4 results. Techniques that failed to execute (2–3 per scenario, consistently across all steps: T1003.003 NTDS dump, T1548.002 UAC bypass, and scenario-specific failures) are excluded from the denominator.

**Table 9**
Step 4 detection coverage ($C_3$) per scenario with 26 KGPT rules (16 process-creation + 10 script-block). $\Delta C_3 = C_3^{\text{strict}} - C_2^{\text{strict}}$ is the primary endpoint.

| Scenario | Exec | $C_3^{\text{strict}}$ | $C_3^{\text{eff}}$ | $C_2$ | $\Delta C_3$ | Pass? |
|---|---|---|---|---|---|---|
| S1-APT29 | 26 | 73.1% | 84.6% | 32.1% | +41.0 pp | ✓ |
| S2-FIN6 | 24 | 70.8% | 79.2% | 46.2% | +24.6 pp | ✓ |
| S3-Sandworm | 28 | 75.0% | 75.0% | 46.7% | +28.3 pp | ✓ |
| S4-WizardSpider | 34 | 67.6% | 79.4% | 40.5% | +27.1 pp | ✓ |
| **Average** | **28** | **71.6%** | **79.6%** | **41.4%** | **+30.3 pp** | **4/4** |

**Table 10**
Step 4 detection outcome classification per scenario. KGPT$^+$ counts techniques where a KGPT rule provided the first or only correctly-mapped detection.

| Scenario | DET | DET$^*$ | TEL | NONE | KGPT$^+$ |
|---|---|---|---|---|---|
| S1-APT29 | 19 (73.1%) | 3 (11.5%) | 4 (15.4%) | 0 | 3 |
| S2-FIN6 | 17 (70.8%) | 2 (8.3%) | 5 (20.8%) | 0 | 5 |
| S3-Sandworm | 21 (75.0%) | 0 | 7 (25.0%) | 0 | 5 |
| S4-WizardSpider | 23 (67.6%) | 4 (11.8%) | 7 (20.6%) | 0 | 4 |

**Primary endpoint: H1 supported at 4/4.** The pre-registered decision rule requires $\Delta C_3 > 0.10$ (10 percentage points) in at least 3 of the 4 primary scenarios. All four scenarios exceed this threshold: $\Delta C_3$ ranges from +24.6 pp (S2-FIN6) to +41.0 pp (S1-APT29), with an average of +30.3 pp. The primary hypothesis is supported.

**Statistical robustness.** For the aggregated results, $C_3 = 71.6\%$ is associated with a 95% Wilson confidence interval of approximately [65%, 78%], while $C_2 = 41.4\%$ yields [36%, 47%], indicating a clear statistical separation between both steps.

**KGPT-attributed detections.** Across all four scenarios, 17 technique×scenario detections are attributable to KGPT rules. Five unique KGPT rules account for these: *Executable Copy to Writable Temp or Admin Share* (T1570, fires in all 4 scenarios), *Network Share Enumeration* (T1135, all 4), *File Rename to Known Ransomware Extension* (T1486, S2/S3/S4), *Credential File Search* (T1552.001, S1), and *Registry Modification of Security-Sensitive Keys* (T1112, S3). The remaining 21 KGPT rules did not fire in this execution but cover technique variants and edge cases documented in the ATT&CK reference for each sub-technique.

**Remaining gaps.** 4–7 techniques per scenario remain in the TEL category after Step 4. These include: process injection (T1055, requires memory-level analysis), data from local system (T1005, where `type` commands are indistinguishable from benign file reads), and several impact-phase techniques (T1485,

T1489, T1529) whose PowerShell invocations through WinRM did not match the authored rule patterns closely enough. Closing these gaps would require either deeper telemetry (ETW providers, memory scanning) or behavioral correlation rules combining multiple weak signals—both beyond the scope of single-technique Sigma rules.

### 7.5. Cumulative Detection Maturity

Table 11 presents the four-step progression, showing the additive contribution of each detection layer. Each row traces one scenario from out-of-the-box SIEM ($C_0$) to full KGPT coverage ($C_3$).

**Table 11**
Cumulative strict detection coverage across the four-step maturity ladder. $\delta_i$ denotes the marginal gain at each step relative to the previous step.

| Scenario | $C_0$ | $\delta_1$ | $C_1$ | $\delta_2$ | $C_2$ | $\delta_3$ | $C_3$ | Total |
|---|---|---|---|---|---|---|---|---|
| S1-APT29 | 21.4% | ±0 | 21.4% | +10.7 | 32.1% | +41.0 | 73.1% | +51.7 pp |
| S2-FIN6 | 19.2% | ±0 | 19.2% | +27.0 | 46.2% | +24.6 | 70.8% | +51.6 pp |
| S3-Sandworm | 30.0% | ±0 | 30.0% | +16.7 | 46.7% | +28.3 | 75.0% | +45.0 pp |
| S4-WizardSpider | 21.6% | ±0 | 21.6% | +18.9 | 40.5% | +27.1 | 67.6% | +46.0 pp |
| **Average** | **23.1%** | **±0** | **23.1%** | **+18.3** | **41.4%** | **+30.3** | **71.6%** | **+48.6 pp** |

Three observations emerge from the ladder:

1. **SigmaHQ rules add zero strict coverage** ($\delta_1 = 0$ in all scenarios). The 2,396 community rules fire prolifically—+33.1 pp in *effective* coverage—but every new alert maps to the wrong ATT&CK technique. For gap-analysis purposes, community rules in their current form do not reduce the count of undetected techniques.
2. **EDR and KGPT rules address complementary technique populations.** EDR averages +18.3 pp on behavioral detections invisible to log-based rules (credential dumping, process injection, binary masquerading), while KGPT rules average +30.3 pp on command-pattern detections (lateral copy, share enumeration, registry modification, ransomware indicators).
3. **KGPT rules produce the largest single-step gain**, despite being the simplest technology layer (26 Sigma rules versus an endpoint agent with kernel-level visibility). This suggests that precisely targeted, correctly mapped rules—derived from systematic gap analysis—are more effective per unit effort than broad-spectrum detection technologies deployed without gap guidance.

The total four-step improvement averages +48.6 pp (from $C_0 = 23.1\%$ to $C_3 = 71.6\%$), demonstrating that a structured maturity ladder can more than triple strict detection coverage from an out-of-the-box SIEM deployment.

## 8. Discussion

### 8.1. Interpretation

The four-step results reveal three structural insights about detection engineering.

**The rule-mapping quality gap is the dominant failure mode.** At Step 2, adding 2,396 SigmaHQ rules produces +33.1 pp in effective coverage but ±0 pp in strict coverage. Community rules consistently fire on the correct malicious activity but annotate it under the wrong ATT&CK technique—typically the execution transport (T1021.006 WinRM) rather than the payload technique (e.g., T1135 Network Share Discovery). This misalignment creates blind spots in technique-level coverage dashboards.

**Log-source path awareness matters more than rule volume.** The KGPT `ps_script` rules demonstrate this: 10 rules targeting PowerShell Script Block Logging (EID 4104) address an entire class of techniques that 2,396 SigmaHQ rules and 6 EDR behavioral rules all missed—not because the detection logic was wrong, but because rules targeted the wrong field (`process.command_line` instead of `ScriptBlockText`). The KG backward traversal makes this diagnosis automatic; without it, the practitioner must recognize the WinRM/ScriptBlock distinction from experience.

**Targeted rules outperform broad-spectrum detection per unit effort.** 26 KGPT rules produce +30.3 pp strict coverage gain—larger than the +18.3 pp from EDR (an endpoint agent with kernel-level visibility) and vastly more efficient than 2,396 SigmaHQ rules (±0 pp strict). This is not a claim that Sigma rules are superior to EDR; rather, that *correctly targeted* rules derived from systematic gap analysis fill gaps that broad-spectrum technologies leave open. The two are complementary: EDR detects behavioral patterns invisible to log-based rules, while KGPT rules provide precise ATT&CK-mapped coverage for command-line and script-block patterns that EDR heuristics do not annotate.

## 8.2. Comparison with Related Work

Winkler and Sharma [13] report 85% overall detection rate for Wazuh against ATT&CK-emulated attacks. Our $C_3 = 71.6\%$ is lower in absolute terms but measures a stricter criterion: only correctly ATT&CK-mapped alerts count. Under our effective coverage metric ($C_3^{\text{eff}} = 79.6\%$), which counts any alert triggered by the technique regardless of mapping, the gap narrows. Importantly, Winkler and Sharma measure a single-pass detection rate without a before/after improvement cycle; our contribution is the measurable $\Delta C$ across detection layers and the diagnosis infrastructure that enables targeted remediation.

The CTID Sandworm and Wizard Spider plans provide the closest methodological prior art with their Detection/Protection two-pass structure. Our results validate the two-pass approach and extend it in three directions: (i) we decompose the single "protection" step into three distinct layers (SigmaHQ, EDR, KGPT) to attribute detection gains, (ii) the KG backward traversal automates the gap diagnosis that the CTID plans leave to manual analysis, and (iii) the pre-registered decision rule ($\Delta C_3 > 0.10$ in $\geq$3/4 scenarios) provides a falsifiable standard rather than a narrative walkthrough.

MITRE Engenuity ATT&CK evaluations [12] benchmark EDR *products* under controlled conditions; our framework benchmarks an *organization's detection configuration* with its specific rule set and telemetry stack. Engenuity evaluations inform product selection; KGPT-style assessments measure how well the selected product is configured for the organization's threat model.

## 8.3. Limitations

**Lab substitutions and the $\Delta$-invariance argument.** Because baseline and post-fix phases run identical substitutions on identical snapshots, every fidelity reduction cancels in the $\Delta C$ comparison. The primary contribution is $\Delta C$, not the absolute coverage values. Key substitutions include: destructive payloads (NotPetya wiper, ransomware binaries) replaced by behavior-equivalent neutered stubs that preserve the same Sysmon/heuristic detection signal; real malware (Exaramel, P.A.S. webshell) replaced by command-surface-equivalent ART tests; Windows Defender real-time protection disabled on all targets (to allow commodity tool execution; AV bypass is out of scope); network egress blocked by a fully isolated VLAN verified per-run. The full substitution table with per-step rationale is published in the artifact repository.

The framework also has the following structural limitations:

- **Scenario selection.** The four scenarios are among the most polished in the CTID library, with above-average SigmaHQ coverage. We report consistency across four motivational profiles and do not claim cross-actor generalisation or geographic diversity.

- **Single SIEM platform.** All measurements use Elastic Security with SigmaHQ rules. Generalisation to Splunk or commercial SIEMs is future work; the methodology is platform-agnostic but the absolute coverage numbers are not.
- **D3FEND coverage gaps.** Not every ATT&CK technique has a D3FEND mapping in the current ontology release. The KG records this as a finding rather than treating it as an error.
- **Operator-as-engineer.** A single PhD student executes the runs, engineers the rules, and analyses the data. The Phase 3 step is unblinded by design: the operator uses gap analysis to write remediation rules. This reflects the intended workflow and is controlled via a fixed *Phase 3 rule-authoring protocol*, requiring all rules to derive from ATT&CK documentation, with justification logging and independent second-rater review.
- **Operator time as a binding constraint.** Full execution of the four primary scenarios at $n = 5$ per phase plus robustness ablation tiers and the production parity sub-run is estimated at ≈95 operator-hours. Operator time was the binding constraint on $n$. Scaling to more actors or more replicates would require additional engineering automation.

### 8.4. Threats to Validity

**Internal validity.** VM snapshot restoration between runs ensures identical starting conditions, but non-deterministic timing in network telemetry may introduce variance in detection delay measurements. We mitigate this with 5 independent runs and non-parametric statistical tests.

**External validity.** Results are obtained in a controlled lab environment with specific SIEM and sensor configurations. Generalization to production environments with different tooling, scale, and noise levels requires further investigation. The use of four scenarios spanning distinct motivational profiles partially addresses this concern.

**Construct validity.** Our six KPIs capture different facets of detection capability but do not encompass all aspects of security operations (e.g., analyst response time, investigation quality). We explicitly scope our measurement to detection engineering, not incident response.

## 9. Conclusion

We presented a knowledge-graph-driven framework for transforming adversary emulation exercises into instruments of measurable detection improvement. A bidirectional knowledge graph encodes the full defensive chain from ATT&CK technique through data component, detection strategy, analytic rule, log source, and sensor, enabling automated gap diagnosis when detection fails.

Evaluated against four threat profiles—nation-state espionage (APT29), financial cybercrime (FIN6), state destructive operations (Sandworm), and big-game-hunting ransomware (Wizard Spider)—the framework demonstrates consistent, substantial detection improvement. The four-step maturity ladder raises strict detection coverage from $C_0 = 23.1\%$ (out-of-the-box SIEM) to $C_3 = 71.6\%$ (with KGPT-guided rules), an average gain of $+48.6$ percentage points. The primary endpoint—the marginal contribution of KG-guided rule engineering alone—averages $\Delta C_3 = +30.3$ pp, exceeding the pre-registered $>10$ pp threshold in all four scenarios.

Two findings are particularly noteworthy. First, 2,396 SigmaHQ community rules produce zero strict-coverage improvement despite massive alert volume, because their ATT&CK annotations systematically map to the execution transport rather than the payload technique. Second, the KG backward traversal identified a structural blind spot—PowerShell techniques delivered via WinRM populate Script Block Logging (EID 4104) but not the process command line targeted by conventional Sigma rules—enabling targeted remediation that a practitioner without graph-guided diagnosis would likely miss.

The framework provides the security community with: (1) a reusable ontology bridging ATT&CK and D3FEND through operational telemetry; (2) a reproducible protocol for purple teaming with formal measurement; (3) empirical evidence that structured knowledge systematically closes detection gaps, with consistent results across four distinct threat profiles.

Future work includes extending the framework to cloud and container environments, scaling to additional threat actors beyond the CTID emulation library, integrating LLM-assisted detection rule generation [19], and applying the methodology to continuous detection validation in production SOC settings.

## 10. Reproducibility

All artifacts required for independent replication are packaged in a repository currently kept private for the review period. They can be provided to the program committee upon request.

## Ethics and Safety Statement

All experiments are conducted on a dedicated, fully isolated virtual laboratory hosted on a private server with no Internet egress, verified per-run via `tracepath` negative-reachability checks. No attacks target real-world systems; every emulated technique executes exclusively within the closed VLAN against purpose-built Windows endpoints and a synthetic Active Directory domain. Destructive samples (NotPetya, Exaramel) are replaced by behavior-equivalent neutered stubs documented in Section 8.3. Credential-dumping artifacts run only against synthetic lab accounts with no relation to real credentials. No production data, user PII, or external network traffic transit the lab at any point. The testbed is destroyed and re-provisioned from clean VM snapshots between scenarios, ensuring no residual artifacts persist across runs.

## A. Example Instrumentation Record

Each Instrumentation Record (IR) is a JSON object with three sections: *offensive* (executor, command, cleanup), *defensive* (expected data components with log source and event IDs, target Sigma analytics, D3FEND countermeasures), and *measurement* (telemetry-presence query, detection-fired query, alert-quality threshold, time window). For example, the IR for S1-APT29 step 5.A.1 (T1003.002, SAM dump via Mimikatz) specifies PowerShell Script Block (EID 4104) and Sysmon Process Access (EID 10) as mandatory data components, three SigmaHQ analytics as detection targets, and a 120-second measurement window. Alert queries are scoped to $[T_{\text{exec}},\ T_{\text{exec}} + \text{window}]$ per-step to prevent temporal pollution. The full JSON schema and 12 worked IRs are published in the artifact repository.

## References


[1] MITRE Engenuity Center for Threat-Informed Defense, Adversary emulation plans, https://github.com/center-for-threat-informed-defense/adversary_emulation_library, 2024. Accessed: 2026-05-13.

[2] Security Risk Advisors, The purple perspective 2026 report, https://www.securityriskadvisors.com/purple-perspective-2026/, 2026. Accessed: 2026-05-13.

[3] S. Zhang, X. Xue, X. Su, Deepop: A hybrid framework for mitre att&ck sequence prediction via deep learning and ontology, Electronics 14 (2025) 257. doi:10.3390/electronics14020257.

[4] Z.-X. Li, Y.-J. Li, Y.-W. Liu, C. Liu, N.-X. Zhou, K-ctiaa: Automatic analysis of cyber threat intelligence based on a knowledge graph, Symmetry 15 (2023) 337. doi:10.3390/sym15020337.

[5] W. Xiong, R. Legany, et al., Cyber security threat modeling based on the MITRE enterprise ATT&CK matrix, Software and Systems Modeling (2022). doi:10.1007/s10270-021-00898-7.

[6] W. Widel, et al., Formalizing attack-defense trees with defensive actions, IEEE Access 10 (2022). doi:10.1109/ACCESS.2022.3197691.

[7] C. Bratsas, et al., Knowledge graphs and semantic web tools in cyber threat intelligence: A systematic review, Journal of Cybersecurity and Privacy (2024).

[8] A. Akbar, F. Rahman, A. Singhal, L. Khan, B. Thuriasingham, The design and application of a unified ontology for cyber security, in: Information Systems Security (ICISS 2023), Springer, 2023, pp. 24–43. URL: https://doi.org/10.1007/978-3-031-49099-6_2. doi:10.1007/978-3-031-49099-6_2, accessed: 2026-05-13.

[9] A. Boyer, E. Dogdu, R. Choupani, J. S. Watson, D. Sanchez, A. Ametu, Toward a unified cybersecurity knowledge graph: Leveraging ontologies and open data sources, in: Recent Advances in Next-Generation Data Science (SDSC 2024), volume 2158 of *Communications in Computer and Information Science*, Springer, 2024, pp. 17–33. URL: https://doi.org/10.1007/978-3-031-67871-4_2. doi:10.1007/978-3-031-67871-4_2, accessed: 2026-05-13.

[10] V. Mavroeidis, S. Bromander, Cyber threat intelligence model: An evaluation of taxonomies, sharing standards, and ontologies within cyber threat intelligence, in: European Intelligence and Security Informatics Conference (EISIC), 2017, pp. 91–98. doi:10.1109/EISIC.2017.20.

[11] V. Mavroeidis, M. Zych, Cybersecurity playbook sharing with stix 2.1, arXiv preprint arXiv:2203.04136 (2022). URL: https://arxiv.org/abs/2203.04136. doi:10.48550/arXiv.2203.04136, accessed: 2026-05-13.

[12] C.-W. Shen, et al., Understanding MITRE engenuity ATT&CK evaluations: An analysis, in: Proceedings of the 19th ACM Asia Conference on Computer and Communications Security (AsiaCCS), 2024. doi:10.1145/3634737.3645012.

[13] D. Winkler, Y. Sharma, Measuring detection capabilities: An empirical study of ATT&CK-based adversary emulation, Computers & Security (2025). doi:10.1016/j.cose.2025.104283.

[14] Rabobank CDC, Dett&ct: Detect tactics, techniques & combat threats, https://github.com/rabobank-cdc/DeTTECT, 2024. Accessed: 2026-05-13.

[15] A. Applebaum, D. Miller, B. Strom, C. Korber, R. Wolf, Intelligent, automated red team emulation, in: Proceedings of the 32nd Annual Conference on Computer Security Applications (ACSAC), 2016, pp. 363–373. doi:10.1145/2991079.2991111.

[16] Red Canary, Atomic red team, https://github.com/redcanaryco/atomic-red-team, 2024. Accessed: 2026-05-13.

[17] R. Uetz, et al., Laccolith: Hypervisor-based adversary emulation with anti-detection, IEEE Transactions on Dependable and Secure Computing (2024). doi:10.1109/TDSC.2024.3374493.

[18] SigmaHQ, Sigma: Generic signature format for siem systems, https://github.com/SigmaHQ/sigma, 2024. Accessed: 2026-05-13.

[19] Security AI Team, Rulegenie: LLM-assisted detection rule optimization, arXiv preprint arXiv:2505.06701 (2025).

[20] Microsoft Security Research, CTI-REALM: A benchmark for end-to-end detection rule generation, arXiv preprint arXiv:2603.13517 (2026).

[21] M. Saint-Hilaire, et al., Matching vulnerability descriptions to D3FEND countermeasures, in: International Conference on Science of Cyber Security (SciSec), Springer, 2024. doi:10.1007/978-981-96-2417-1_7.

[22] Russinovich, Mark and Garnier, Thomas, Sysmon - sysinternals, https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon, 2026. Accessed: 2026-05-13.

[23] Elastic, Elastic security solution & project type overview, https://www.elastic.co/docs/solutions/security, 2026. Accessed: 2026-05-13.

[24] SigmaHQ, pysigma: Python library to parse and convert sigma rules into queries, https://github.com/SigmaHQ/pySigma, 2026. Accessed: 2026-05-13.

[25] SigmaHQ, pysigma-backend-elasticsearch: pysigma elasticsearch backend, https://github.com/SigmaHQ/pySigma-backend-elasticsearch, 2026. Accessed: 2026-05-13.

[26] A. Virkud, W. Liu, A. Gaur, G. Lynch, C. Kanich, C. R. Taylor, Evaluating EDR telemetry with MITRE ATT&CK, in: Proceedings of the 33rd USENIX Security Symposium, 2024.