Introducing APEX

Adversarial Pattern Extraction and Correlation

Cribl®

The **AI Platform** for Telemetry

A new detection framework for the AI-enabled threat era

The Pyramid of Pain has been industry vocabulary for over a decade. Everyone knows the apex is where defenders should operate. Almost no one has built a systematic framework for actually getting there until now.

APEX, Adversarial Pattern Extraction and Correlation, is a behavioral detection model designed from first principles for an era in which AI has made the bottom of the pyramid worthless. Hashes, IPs, and domains are no longer durable detection surfaces. They are disposable inputs that a sufficiently capable attacker — or a sufficiently capable AI — can rotate faster than any feed can publish them. APEX abandons the race to keep up with attacker infrastructure and builds detection at the one layer attackers cannot change: behavior.

This session introduces the APEX framework and the empirical foundation behind it. Nicole Beckwith will present findings from the Anthropic Frontier Red Team's analysis of 832 real-world threat actors — the largest dataset of AI-enabled attacker behavior published to date — and show what the data actually reveals about how high-risk actors operate. The insight that anchors APEX: individual ATT&CK techniques are weak signals. Low-risk and high-risk actors use many of the same techniques. What separates a commodity actor from a fully autonomous AI attack platform is not which techniques appear, it is which techniques appear together, in what sequence, within what time window. APEX operationalizes that insight into a detection architecture built on behavioral chains, not behavioral indicators.

The session will cover the three pillars of APEX: extraction — identifying adversarial technique sequences from heterogeneous telemetry without depending on static normalization or schema reserialization; pattern — correlating technique chains against empirically-derived risk weights calibrated to real attacker archetypes; and correlation — closing the detection loop through agentic investigation that produces analyst-ready context in under 90 seconds, with zero human involvement before the decision point.

APEX is also a framework for evaluating your existing detection program. Attendees will leave with a clear diagnostic: where their current detection logic is exposed to indicator decay, where telemetry gaps are silently suppressing behavioral coverage, and which pipeline architecture decisions are either enabling or undermining the shift to apex-level detection.

This is not a product announcement. It is the introduction of a detection philosophy built for a threat landscape that has already changed — by a practitioner who has spent over a decade engineering detection programs that actually have to work when the attacker is already inside.

APEX: because the top of the pyramid was always the destination. ***Now there is a map.***

CONTENTS

02 Abstract

03 About the authors

05 1. The problem: AI has broken the IOC Model

07 2. Why this matters now: The empirical evidence

10 3. The APEX model: Architecture and theory of detection

14 4. Why behavioral detection requires the pipeline

21 5. The APEX test detection library: 23 behavioral chains

24 6. AI Signal/detection authoring

29 7. Conclusion: A detection program that survives the AI era

30 8. References

TECHNICAL WHITE PAPER

APEX

Adversarial Pattern Extraction and Correlation

A Signal-Based Behavioral Detection Framework for the AI-Enabled Threat Era

Nicole Beckwith
Senior Director, Security Engineering & Operations
Cribl

Nate Zemanek
Staff Solutions Engineer
Cribl

June 2026

ABSTRACT

The traditional indicator-of-compromise (IOC) detection paradigm is in terminal decline. The emergence of generative artificial intelligence as an attacker capability has industrialized the production of polymorphic malware, disposable phishing infrastructure, and bespoke offensive tooling at a pace that no indicator feed can match. The bottom layers of the Pyramid of Pain, hashes, IPs, domains, have been reduced to perishable commodities. The security operations center (SOC) that still anchors its detection program to IOC matching is fighting last decade's war with last decade's weapons.

This paper introduces the APEX Model, Adversarial Pattern Extraction and Correlation, a behavioral detection framework purpose-built for the AI threat era. APEX treats individual MITRE ATT&CK techniques as atomic building blocks, correlates their sequencing and timing across the kill chain, and applies deep time series forecasting to score behavioral clusters against empirically derived risk weights. The result is a detection architecture that remains durable as attacker tooling mutates, because it targets what attackers must do, not what tools they happen to use.

This framework is informed by the Anthropic Frontier Red Team's analysis of 832 real-world threat actors and 13,873 TTP observations collected between March 2025 and March 2026, the most comprehensive empirical dataset of AI-enabled cyber threat behavior published to date. APEX's test detection library covers all 14 MITRE ATT&CK tactic categories identified in that dataset and is specifically calibrated against the ARiES actor risk scoring methodology to ensure highest-priority detections map to the highest-risk behavioral signatures.

ABOUT THE AUTHORS

Nicole Beckwith - Author

Senior Director, Security Engineering & Operations | Cribl

Nicole Beckwith brings over a decade of hands-on leadership in Security Operations Center (SOC) engineering, detection architecture, and threat intelligence to the challenge of behavioral-based security at scale. She has spent her career at the intersection of detection engineering, data pipeline architecture, and adversary tradecraft, designing the detection programs that enterprise and government defenders actually rely on when the alert queue runs deep and the attacker is already inside.

Nicole is widely recognized as a practitioner-leader in the SOC space: someone who has built and run detection teams from the ground up, written the detection logic that catches real intrusions, and sat through the midnight investigations that reveal exactly where conventional tooling fails. Her work spans SIEM engineering and architecture, incident response, threat intelligence, threat hunting, telemetry design, cloud-native security architectures, insider risk and UEBA program development, and most recently the application of agentic AI to the detection and investigation workflow.

At Cribl, Nicole leads security strategy with a focus on how the data pipeline layer can be elevated from a passive log-forwarding tier into an active participant in the detection lifecycle. The APEX framework is the product of that conviction: that meaningful behavioral detection cannot be bolted onto a system that wasn't designed for it, and controlling the data before it becomes an event is the structural advantage the industry has been missing.

Nicole is the co-author of the CTI-CMM (https://cti-cmm.org/ 2024) and the author of the predecessor white paper "From Hashes to Hunts: How AI Is Forcing Detection to Climb the Pyramid" (April 2026) and the internal executive briefing "The Problem No One Has Actually Solved" (June 2026), both of which informed the APEX model.

She holds deep expertise in MITRE ATT&CK framework application, threat actor behavioral profiling, large-scale telemetry normalization, and the operational mechanics of high-fidelity detection programs.

> *Nicole is a SOC veteran who stopped waiting for the SIEM industry to deliver behavioral detection and built it herself.*

Nate Zemanek – Co-Author

Staff Solutions Engineer | Cribl

Nate Zemanek is a Staff Solutions Engineer at Cribl with over two years of experience helping enterprise customers operationalize telemetry pipelines at scale across Cribl Stream, Search, and Lakehouse. His background spans the full observability and security stack: prior to Cribl, he spent three years at Panther focused on SIEM architecture and detection engineering, and earlier at Duo Security, where he developed foundational expertise in security and IT.

Before transitioning into solutions engineering, Nate held software development and business automation roles in the pharmaceutical industry, giving him a practitioner's perspective on the operational constraints and compliance requirements that enterprise teams navigate daily. This cross-disciplinary foundation spanning application development, security operations, and large-scale data pipeline design informs his approach to architecting solutions that are both technically rigorous and operationally sustainable.

When he's not spending time with his wife and two daughters, he's probably vibe coding and automating something that doesn't need to be automated.

> *Nate is an engineer who saw the promise of behavioral detection and built the solution.*

1. The Problem: AI Has Broken the IOC Model

1.1 Two Decades of Detection Muscle Memory

For most of the history of network defense, the question “what does an attack look like?” had a simple answer: it looks like a hash, an IP address, a domain, a URL, a registry key. Indicators of compromise (IOCs) gave defenders something concrete to look for and something concrete to share. They underpinned antivirus, intrusion detection, threat intelligence platforms, and the entire commercial threat-feed industry. Two decades of detection muscle memory was built around them.

The IOC lineage runs back to the earliest commercial antivirus products of the late 1980s, where detection meant matching byte sequences against a signature database. Network-based detection followed in the late 1990s with Snort, which let defenders write rules against packet content. The 2010s formalized the IOC as a unit of exchange: Mandiant published OpenIOC in 2011, MITRE released STIX and TAXII the following year, and the ecosystem of ISACs, commercial feeds, and open-source platforms like MISP turned IOC sharing into a global industry. The implicit promise was economically efficient: an attacker burns infrastructure on victim A, victim A publishes the artifacts, every other defender blocks them. Detection could be a lookup.

1.2 The Pyramid of Pain, and Why Defenders Lived at the Bottom

In 2013, David Bianco published the Pyramid of Pain, which has aged into arguably the most important foundation for the field. From the bottom up: hash values, IP addresses, domain names, network and host artifacts, tools, and at the apex, tactics, techniques, and procedures (TTPs). The point of the pyramid was not that low-level indicators are useless, it was that they cost an attacker almost nothing to change. Burning a hash means recompiling. Burning an IP means renting another VPS. Burning a TTP means redesigning the operation.

For most of the 2010s, defenders lived at the bottom of the pyramid because that was where the tools, feeds, and economics pointed. The cost of that choice was visible in dwell-time statistics that stubbornly refused to fall, and in major intrusions where every IOC was novel. The detection program was optimized for the wrong tier of the pyramid, and the industry largely knew it, but the alternative required a fundamentally different data pipeline, different tooling, and a different theory of how detections should work.

1.3 AI Compresses the Decay Curve to Zero

Even before generative AI, IOCs were under pressure. Fast-flux DNS, domain generation algorithms, commodity bulletproof hosting, and rotating command-and-control infrastructure had already shortened the useful life of network indicators to hours in some campaigns. Hash IOCs survived a little longer for commodity malware and not at all for targeted intrusions, where one-off implants and living-off-the-land binaries dominated.

AI compresses this curve sharply. Three patterns are now empirically visible across real-world campaigns.

- **Pattern 1:** Polymorphic and metamorphic malware at commodity cost. LLMs and code-generation tools make it trivial to regenerate a loader per victim or per beacon, defeating hash IOCs and most static signatures by construction. The marginal cost of a unique binary is now approximately the marginal cost of a model inference.

- **Pattern 2:** Mass-produced phishing and credential-harvesting infrastructure. Cheap domain registration combined with LLM-generated lure content and AI-generated brand-impersonation pages defeats both content-pattern matching and the human reviewers who once flagged obvious typos. Domain IOCs become single-use; by the time a feed publishes one, the campaign is already on the next thousand.

- **Pattern 3:** Bespoke offensive tooling on demand. What once required a developer on the operator's payroll can now be produced in an afternoon. Tool-level IOCs, imphashes, YARA rules tuned to specific code patterns degrade accordingly.

The aggregate effect is that the bottom three layers of the Pyramid of Pain have become a perishable input rather than a durable detection surface. I want to be clear though, the IOC feed is not dead, it remains useful for blocking known-bad at volume, for alert enrichment, and for retroactive hunting once a campaign is disclosed. What is gone is its role as the primary detection mechanism.

> *The defensive answer is not a smarter IOC feed. It is a center-of-gravity shift up the pyramid and that requires a completely different detection architecture.*

2. Why This Matters Now: The Empirical Evidence

2.1 The Anthropic Frontier Red Team Dataset

In June 2026, the Anthropic Frontier Red Team published the most comprehensive analysis of AI-enabled cyber threats to date. The full report and interactive ATT&CK Navigator are available at:

- https://www.anthropic.com/news/AI-enabled-cyber-threats-mitre-attack
- https://www.anthropic.com/research/attack-navigator

The dataset covers 832 real-world threat actors, 13,873 individual TTP observations, and a 12-month collection window (March 2025 through March 2026).

Several findings from this dataset are directly foundational to the APEX models' initial ruleset:

- **Evasion is universal:** Defense evasion dominated at 84.4% of all actors, making it the single most prevalent MITRE ATT&CK tactic category, and confirming that signature-based detection is the primary target of attacker counter-measures.

- **AI-assisted tooling is mainstream:** Capability development (T1587/T1588) was observed in 69% of actors, indicating that AI-assisted malware generation is now a baseline attacker behavior, not an advanced-tier capability.

- **Directional shift into post-access:** Account Discovery (T1087) increased 8.9% and Automated Exfiltration (T1020) increased 6.2% between the first and second halves of the study period, representing the clearest directional shift: attackers are moving AI assistance deeper into live, post-compromise operations.

- **Chaining predicts risk:** Lateral movement was the single strongest predictor of a high ARiES risk score, elevating average actor risk by 10.5 points, meaning the behavioral chain matters far more than any individual technique.

2.2 The GTG-1002 Archetype: The Highest-Risk Actor Pattern

The Anthropic dataset surfaces a threat actor archetype designated GTG-1002, which achieved the maximum ARiES risk score of 100. GTG-1002's kill chain is the empirical basis for APEX's highest-severity detection tiers and merits specific examination because it represents the realistic upper bound of what AI-enabled adversary operations currently look like.

GTG-1002 weaponized Claude Code running on a Kali Linux machine, integrating open-source penetration testing tools as MCP (Model Context Protocol) servers, effectively turning the AI into an autonomous attack platform. The AI executed commands, made real-time tactical decisions, and chained reconnaissance through exploitation through lateral movement through exfiltration with minimal human input at each stage.

Key behaviors in the GTG-1002 kill chain that APEX specifically targets:

- **Autonomous recon at machine speed:** Autonomous network scanning and mapping of dozens of internet-facing services during reconnaissance, then discovery of internal admin portals, databases, logging servers, and workflow systems once inside the network, all executed at machine speed within a 15-minute window that no human operator could replicate.

- **SSRF-to-cloud-metadata pivot:** SSRF vulnerability exploitation in a public-facing web server to proxy commands into an internal cloud environment, followed by credential harvesting from the cloud metadata service (169.254.169.254) to enable lateral movement.

- **Cloud credential harvesting:** SSH private key and service account token harvest from AWS Secrets Manager and cloud metadata services, followed by IAM manipulation to escalate privileges and persist.

- **Automated exfil with evidence destruction:** Web shell deployment on internally reached servers, followed by automated bulk exfiltration using scripted loops, the staging-and-wipe pattern designed to minimize forensic evidence.

> *The Anthropic report explicitly states that agentic scaffolding, not technique breadth, is the true differentiator of the highest-risk actors. GTG-1002 did not use more techniques than lower-risk actors. It used AI to chain them faster, more autonomously, and with greater operational security than human-driven attacks.*

2.3 Why Traditional SIEM Architecture Fails Against This

Legacy SIEMs and even modern AI-augmented platforms share five structural failures that the GTG-1002 archetype exploits by design:

- **Failure 1:** Difficult cross-source correlation across unnormalized logs. OCSF was invented to normalize data to work better together, but in the case of detection workflows, applying it rigorously is often over-engineering at best and prohibitively complex at worst. Data normalization exists to work within the parameters of a one-dimensional detection framework, not the multi-dimensional framework we now need.

- **Failure 2:** Static detection in a dynamic threat landscape. Sigma rules and ML models are trained at a point in time. Attackers iterate in days; AI iterates in milli-seconds; detection logic iterates in quarters.

- **Failure 3:** Correlation without causation. Traditional correlation engines connect events but do not reason about why those events matter in the context of this specific environment, this specific user, at this specific moment in the kill chain.

- **Failure 4:** No feedback loop. When a detection misfires, is too noisy, too late, or completely absent, there is no mechanism for the system to learn. Institutional knowledge exists entirely in the heads of analysts who eventually leave.

- **Failure 5:** Detection and data pipeline as separate kingdoms. The SIEM sits downstream of the data pipeline. By the time data reaches the detection layer, the ability to enrich it, reshape it, or route it intelligently based on threat context is already gone.

The result is alert fatigue so severe that the average SOC ignores or dismisses over 45% of all security alerts not because analysts are inadequate, but because the architecture produces noise at a rate that exceeds human cognitive capacity. APEX is designed from first principles to address all five of these structural failures simultaneously.

3. The APEX Model: Architecture and Theory of Detection

3.1 What APEX Stands For

APEX is an acronym that encodes the four architectural pillars of the framework:

A, Adversary	Detection anchored at the apex of the Pyramid of Pain targeting the adversarial objectives and behavioral sequences attackers cannot change by rotating infrastructure. APEX detects what they must do, not what tools they happen to use.
P, Pattern	Behavioral chains as the fundamental unit of risk. Individual ATT&CK techniques are weak signals, low-risk and high-risk actors use many of the same techniques at similar rates. Risk emerges from which techniques appear together, in what sequence, and within what time window. The Pattern layer identifies these chains against empirically-derived risk weights.
E, Extraction	Signal-based behavioral detection that normalizes the conclusion, not the data. APEX extracts purpose-built signal records from raw vendor telemetry preserving ground truth while producing a minimal, ATT&CK-mapped output that chain correlation can operate against regardless of source data shape. Vendor format changes become query fixes, not unrecoverable mapping errors.
X, Correlation	Cross-entity TTP correlation powered by time series forecasting. The framework establishes behavioral baselines across heterogeneous telemetry streams, scoring deviations against the forecasted normal. The Correlation layer closes the loop: from first behavioral signal through agentic investigation to analyst-ready brief in seconds.

3.2 Adversarial Layer: ATT&CK as the Detection Language

MITRE ATT&CK has become the shared language of behavioral detection engineering. APEX uses it as its primary schema, not as a compliance checklist, but as the formal grammar for expressing attacker intent. Every detection in the APEX library is expressed as a sequence of ATT&CK techniques, which means:

- Detection coverage is auditable at the tactic and technique level, not just as a rule count. A CISO can answer "do we detect T1021 (Remote Services) in the context of post-access lateral movement?" with a specific yes, no, or partial, and understand the gap.

- Detections generalize across vendor telemetry. A behavioral chain defined in ATT&CK terms can fire against CrowdStrike telemetry, against Palo Alto NGFW logs, against AWS CloudTrail, against a bespoke syslog feed.

- New attacker techniques can be added to existing detection chains without redesigning the entire detection. If a new post-access persistence mechanism emerges, it can be expressed as a new ATT&CK sub-technique and inserted into the relevant APEX detection chain with minimal engineering effort. Think of these as building blocks.

3.3 The Pattern Layer: TTP Sequences as Risk Units

The core insight of APEX, and the place where it diverges fundamentally from legacy SIEM architectures, is that individual MITRE ATT&CK techniques are weak signals. The Anthropic dataset confirms this empirically: low-risk and high-risk actors use many of the same ATT&CK techniques at statistically similar rates. What separates a commodity actor from the GTG-1002 archetype is not which techniques appear, but which techniques appear together, in what sequence, within what time window, and with what co-occurrence strength.

APEX's pattern layer treats each MITRE ATT&CK technique as an atomic 'building block' and applies two ordering constraints that transform them into meaningful detection signals:

- **Ordering:** Sequence constraint: techniques must appear in a kill-chain-consistent order within the correlation window. T1087 (Account Discovery) followed by T1003 (Credential Dumping) followed by T1560 (Archive Collected Data) is a post-compromise staging chain. T1003 appearing in isolation is an ambiguous signal that produces alert fatigue.

- **Timing:** Timing constraint: the correlation window is derived from real-world attack data, not arbitrary engineering choices. The 15-minute window for autonomous network reconnaissance (Detection 11 in the inaugural list) is calibrated to the speed differential between AI-automated scanning and human-directed recon, a speed that cannot be achieved manually and therefore filters out legitimate administrative behavior.

> ***Key principle:*** *Detections should be sequenced and chained. Individual techniques are weak signals, low-risk and high-risk actors use the same techniques at similar rates. Risk emerges from which techniques are combined, where they appear in the kill chain, and how fast they execute.*

The implementation mechanism for the pattern layer is the signal: a small, purpose-built record asserting that vendor-specific evidence of an ATT&CK technique was observed for a specific entity at a specific time. Rather than requiring full event normalization at ingest before detection logic can fire, the signal model normalizes the conclusion, producing ATT&CK-mapped, entity-centric outputs from vendor-specific query logic. Detection chains operate against this normalized signal layer regardless of the underlying telemetry shape. Section 4 details the two-tier query architecture, signal schema, and enrichment model that make this work in practice.

3.4 The Extraction Layer: Signal Foundation and Forecasting

The "Extraction" layer of APEX draws on the ARIES framework for deep time series forecasting, specifically the property assessment and model recommendation system described in the ARIES research paper (Wang et al., 2025, arXiv:2509.06060). ARIES establishes a systematic relationship between the six statistical properties of a time series (trend strength, seasonality, volatility, memorability, heteroscedasticity, anomaly density) and the deep forecasting model best suited to capture those properties.

Additionally calling on the ARiES framework Anthropic created and outlines in their posts. “ARiES is a composite score built from three signals: the actor’s threat profile, the model’s contribution to the requested harm, and the observed or potential impact, using addition rather than multiplication to come to the final number. The higher the score, the higher-risk the AI enabled actor is.”

Applied to security telemetry, this is a non-trivial distinction. Process event streams, authentication logs, network flow data, and cloud API audit logs each exhibit fundamentally different statistical properties. These models help provide the principled framework for making decisions on time + risk rather than applying a single forecasting architecture across heterogeneous behavioral signals.

In the APEX context, the extraction layer serves two functions: it establishes behavioral baselines per entity (user, host, service account, workload) across each relevant telemetry dimension, and it flags deviations from those baselines as candidate signals for the correlation layer. The forecasting component provides the 'what is normal' ground truth against which behavioral anomalies are scored within a set timeframe.

3.5 The Correlation Layer: Agentic Closure

The Correlation layer of APEX addresses what happens before and after a behavioral chain fires. In a legacy SIEM architecture, the answer is: an alert appears in a queue. An analyst, 47 alerts deep, picks it up and attempts to reconstruct why the event was suspicious, without the context that was available at the moment of detection.

APEX closes this loop through agentic investigation. When a behavioral chain fires, an AI agent immediately initiates an autonomous investigation workflow:

- Profiles the user's or host's baseline behavior over a 30-day lookback window
- Checks whether destination assets are classified as sensitive in the asset inventory
- Queries the identity provider for MFA status, device enrollment state, and recent authentication anomalies
- Assesses peer group deviation, asking “Is this behavior normal for this user's role and department?”
- Cross-references threat intelligence for the specific TTP combination observed
- Evaluates blast radius, what other assets are reachable from the compromised entity?

- Generates a priority-ranked investigation brief with supporting evidence and recommended response actions, in language an analyst can act on immediately

The result is that the analyst receives a brief, not a raw alert. Time from first behavioral signal to actionable investigation context: typically under 90 seconds. Analyst involvement before the decision point: zero. This is the operational reality that the SOC has been promised by the behavioral detection vendor community for over a decade, and that the architectures of those vendors have consistently failed to deliver.

4. Why Behavioral Detection Requires the Pipeline

4.1 Detection Is Hungrier Than IOC Matching

Behavioral and TTP-based detection is significantly more telemetry-intensive than IOC matching. An IOC engine needs to compare observed values against a list. A behavioral engine needs full process telemetry, command-line arguments, network flow metadata, identity events, cloud audit logs, and DNS, ideally correlated, and delivered with sub-minute latency. For a medium enterprise, that is multi-terabyte-per-day territory. For a large one, it is an order of magnitude more.

This creates a hard dependency: behavioral detection only works at scale when you control the data before it becomes an event. The telemetry pipeline is not a plumbing layer underneath the detection architecture, it is part of the detection architecture. The relevant questions are no longer just 'where do we send this log?' but:

- Which fields does each ATT&CK technique's detection logic actually require, and are those fields present before the data leaves the source?
- What gets enriched in motion, asset classification, identity context, threat intelligence, before reaching the detection layer?
- What gets routed to hot detection versus cold storage versus replay queues?
- How is data shaped so that a behavioral detection written once can fire across the entire estate?

4.2 The Two-Tier Query Model

The APEX model operates through two tiers of scheduled Cribl Search queries, calibrated to where detection logic lives relative to the data shape it needs:

- **Tier 1, Signal queries:** Signal queries run against raw vendor datasets such as CrowdStrike FDR, Palo Alto NGFW, CloudTrail, Okta, and others, on a scheduled cadence. Each signal query encodes vendor-specific logic addressing field names and key-value pairs exactly as they exist in source events. No normalization layer sits between the query and the data. Each signal query covers one MITRE ATT&CK technique in one source. On a match, it emits a record in the common signal schema and exports it to a single Lakehouse-backed signal dataset via Cribl Search's export operator. This one signal dataset is simultaneously the normalization layer as well as the correlation layer.
- **Tier 2, Detection queries:** Detection queries run against the signal dataset only. Each detection implements an APEX behavioral chain: N techniques, in kill-chain-consistent order, within a correlation window, joined on entity. Because all signals share one schema, detection queries are vendor-agnostic by construction. A chain written once fires whether the constituent

signals came from CrowdStrike, Palo Alto, or CloudTrail. A detection hit produces an alert with the full contributing signal chain as evidence.

This two-tier model preserves every core APEX claim, chain over point, ATT&CK as grammar, source-agnosticism, while delivering a critical operational property: raw events are never discarded or mutated. Signal queries run against preserved source data, so a corrected query can regenerate historical signals. Every mapping error is recoverable. The contrast with ingest-time normalization is fundamental: when data is transformed at ingest and the transformed copy becomes the system of record, mapping errors are baked into the data at rest and retroactive correction requires replaying raw data through corrected pipelines at significant cost, if raw data was even retained.

The two-tier model also preserves cost-aware detection. Lightweight signal queries – admin actions, suspicious parent-child process pairs, rare service binaries executing from non-standard paths – run at high frequency against the raw stream and route potentially suspicious behavioral clusters directly to the investigation pipeline. Expensive chain evaluations requiring long lookback windows or cross-entity graph traversal run against the signal Lakehouse, and can also do API context enrichment with your existing products. The distinction ensures the data licensing model does not become the ceiling on detection fidelity.

4.3 The Signal Schema: Normalizing Conclusions, Not Data

The core insight of the APEX signal model: behavioral detection does not require normalized events. It requires normalized signals. **A signal is a small, purpose-built record that captures the kinds of events which, in the right context and sequence for a given entity, may suggest a MITRE ATT&CK technique is in play. On its own it is only a pixel, but in combination it helps reveal the larger behavioral picture.** The raw data stays raw. Vendor-specific detection logic lives in these signal queries, where it is versionable, testable, and instantly correctable.

The signal schema is deliberately minimal, approximately 15 fields versus hundreds in a full OCSF event class. Key design decisions:

- **Entity model:** Entities are typed values with roles (actor, source, destination, target). Detection-time joins chain signals by matching entity type and value across techniques. Roles let chains express directionality, lateral movement where host A's destination becomes host B's source in the next technique.

- **Dual timestamps:** event_time drives correlation windows. detection_time exists to measure signal latency and debug scheduling gaps. These must never be conflated, a signal emitted at 14:00 for an event at 11:00 should correlate at 11:00 in the chain window, not 14:00.

- **Confidence scoring:** Confidence per signal accounts for telemetry quality variation. EDR process telemetry warrants higher confidence than inference from firewall logs. Detection chains can require minimum aggregate confidence across the contributing signal set.

- **Enrichment is additive:** Context enrichment is optional and non-blocking. Enrichment failure or staleness must never suppress a signal. Null context de-enriches; it does not de-detect.

- **Evidence traceability:** evidence.fields carries the raw key-value pairs that fired the signal logic. Analysts see vendor-native evidence, not a normalized abstraction. source_event_ids and raw_query_ref make every signal traceable back to ground truth.

The raw event is ground truth. Any architecture that discards vendor semantics in favor of a derived representation has made its worst-case failure mode unrecoverable. The signal model preserves ground truth and normalizes only the conclusion.

4.4 Enrichment: Three Tiers

Enrichment serves a singular architectural objective: environmental context. It transforms a signal or detection into an actionable decision tree by addressing critical triage questions: is this observation a benign anomaly, and how does the business criticality of the involved entities recalibrate the risk score? The ground truth required to answer these questions is siloed within disparate systems of record: CMDBs for asset hierarchy, IAM platforms for identity posture, and ITSM tools for operational state.

For a decade, high-fidelity context has been the "white whale" of detection engineering, the missing variable in the signal-to-noise equation that has historically overwhelmed SOC analysts. By leveraging federated search to query these sources in real time, we can architect a more durable approach to behavioral context.

While the long-term architectural trajectory may eventually render detection-layer enrichment obsolete through autonomous triage agents, the current economic reality of high-fidelity coverage presents a significant constraint. Scaled detection libraries naturally produce a high volume of candidate signals, and so relying exclusively on agentic workflows for every firing event currently incurs a prohibitive inference cost. The usage of skip-logic here is not always a compromise, as the most critical triage questions often possess binary or highly structured answers that are efficiently resolved within the KQL layer itself.

Where a traditional pipeline enriches events in motion at ingest time, the signal-based model enriches at signal time. Enrichment cost scales with signal volume, not telemetry volume, which preserves the same detect-before-you-pay economics at one layer up the stack. Three enrichment tiers provide different cost and freshness profiles:

- **Tier E1, Live API:** Live API queries join directly against API-backed datasets at signal execution time. Reserved for low-frequency, high-severity paths: a Tier 3 chain candidate justifies a synchronous identity-posture lookup; a commodity signal firing hundreds of times

per hour does not. API latency lands in the query's critical path and rate limits are shared with other consumers.

- **Tier E2, Scheduled lookups:** Scheduled lookup generation is the workhorse. Scheduled searches query APIs on a cadence matched to how fast the source changes and materialize the results as lookup tables via Cribl Search's export to lookup operator. Cadence guidance: asset inventory hourly to daily; identity posture every 15 to 60 minutes; threat intelligence per feed cadence. Signal queries then enrich with a plain lookup operator at effectively zero marginal cost.

- **Tier E3, AI-curated risk lookups:** AI-curated lookup tables are the novel tier. An AI skill, combining the Cribl MCP server with direct API access, regularly queries source systems, evaluates each entity against a written risk rubric (exposure, privilege concentration, patch posture, recent signal history, peer-group deviation), assigns a risk profile with explicit rationale, and writes the result as a lookup table. The distinction from Tier E2 is synthesis: E2 copies facts an API already asserts (this user has MFA enrolled); E3 produces conclusions no single field carries (this host is high risk because it is internet-facing, running services with known-stale patch levels, and has had three privileged accounts authenticate to it this week). AI judgment is materialized as data and consumed at lookup speed with zero inference latency in the detection hot path.

4.5 Governance: AI-Curated Enrichment

The AI curation approach in Tier E3 carries specific governance requirements that distinguish it from standard enrichment. The same principles that define trustworthy agentic investigation (Section 3.5) apply to agentic enrichment:

- **Audit trail:** Provenance is mandatory. Every row carries rationale, generated_at, and generated_by. When a detection escalates in part because a lookup said 'high risk,' the analyst must be able to see the reasoning that produced that conclusion, not just the conclusion itself.

- **Drift visibility:** Reclassification is visible. previous_risk_level and change_reason columns, plus a per-run change report, make AI reclassification auditable. A skill silently downgrading 40 hosts should be caught in review, not discovered mid-incident.

- **TTL enforcement:** Staleness is explicit. A ttl_hours field lets consuming queries treat expired rows as null context rather than trusting stale AI judgment.

- **Failure posture:** Failure de-enriches, never suppresses. A missing, stale, or malformed enrichment row removes context from a signal, it never prevents the signal from being emitted or the detection from firing.

- **Feedback input:** Analyst dispositions feed back in. True positive and false positive outcomes on the alert dataset become inputs to the next curation pass, recovering the closed learning loop at the enrichment tier.

4.6 Implementation Path: OCSF or Signal-Based?

OCSF (Open Cybersecurity Schema Framework) and ECS (Elastic Common Schema) remain valid normalization strategies for organizations that have already invested in them, or for use cases where full-event normalization provides independent value, compliance, long-term retention search, cross-product analytics. The two approaches are not mutually exclusive. Signal queries can run against raw or OCSF-normalized data with equal effectiveness.

The table below summarizes the operational trade-offs between the two implementation paths. The choice depends primarily on whether full-event normalization already exists in the environment and whether its failure modes are acceptable given the organization's raw data retention posture.

Property	Ingest-Time OCSF Normalization	Search-Time Signal Model
Error recovery	Requires raw data replay, often impossible if raw was not retained	Re-run corrected query over raw data; history is regenerable
Onboarding cost per source	Full schema mapping across all event fields	One query per MITRE technique of interest; unused fields ignored
Vendor format drift	Silent mapping breakage baked into data at rest	Query fix deployed in minutes; historical signals regenerable
Data at rest	Transformed, lossy, vendor semantics may be lost or distorted	Raw, ground truth preserved; signals are additive metadata
Normalized surface	Hundreds of fields per OCSF event class	~15 fields per signal record
Detection vendor-agnosticism	Achieved at event level, one schema for all sources	Achieved at signal level, one schema for all signals

Analyst evidence quality	Normalized field names, abstracted from vendor events	Vendor-native key-value pairs in evidence block, directly actionable
Existing OCSF investment	Directly compatible, this is the native path	OCSF events can coexist; signal queries run against raw or normalized data

4.7 The Closed-Loop Feedback Architecture

One of the five structural failures of legacy SIEM architecture identified in Section 2.3 is the absence of a feedback loop. When a detection misfires, too noisy, too late, or completely absent, there is no mechanism for the system to learn.

The pipeline-native behavioral detection architecture enables a closed-loop feedback model that no downstream SIEM can replicate. In the signal-based implementation, the feedback paths are explicit and bidirectional:

- When the agentic investigation layer determines that a detection fired correctly, that outcome signal flows back into the pipeline to reinforce the routing, enrichment, and suppression rules that produced the true positive. True-positive analyst dispositions also feed the next E3 AI curation pass, calibrating risk scores toward confirmed attacker behavior.

- When a detection fires incorrectly, the false positive signal modifies the pipeline-layer pre-filters for that specific behavioral pattern, reducing noise at the source rather than adding suppression logic downstream. False positive dispositions feed back into signal confidence weights and enrichment thresholds.

- When the agentic layer identifies a behavioral pattern that did not fire an existing detection, but that investigation context reveals it as malicious, that pattern can be expressed as a new signal query and detection chain, deployed without waiting for a SIEM rule cycle.

> *The pipeline learns. This is a closed loop that no SIEM architecture built downstream of the telemetry collection layer can replicate, because the feedback cannot reach the data before it becomes an event.*

Capability	Legacy SIEMs	APEX / Pipeline-Native
Data quality at detection	Post-ingestion, degraded	Pre-ingestion, enriched in motion
Detection adaptability	Manual tuning cycles (quarterly)	Continuous learning loop
Pipeline feedback	None	Native, bidirectional
Multi-source normalization	Schema-dependent	Signal-boundary agnosticism
Cost model	Index everything to detect	Detect before indexing; enrich at signal volume
Time to investigation context	Minutes to hours	Under 90 seconds
Analyst involvement at triage	Required for every alert	Zero until decision point
Error recovery	Manual re-ingestion (if raw retained)	Re-run corrected signal query over preserved raw data

5. The APEX Test Detection Library: 23 Behavioral Chains

5.1 Design Principles

The initial APEX detection library used in testing is a set of 23 behavioral detection chains covering all 14 MITRE ATT&CK tactic categories represented in the Anthropic Frontier Red Team dataset. Each detection is designed according to four principles:

- **Chain over point:** No detection fires on a single technique. Every detection requires a sequence of several correlated TTPs observed in sequence within a defined time window.
- **Risk-calibrated windows:** Correlation windows are derived from empirical attack data, not engineering convenience. The 15-minute window for autonomous reconnaissance reflects the AI-speed differential that distinguishes automated scanning from human-directed discovery.
- **Response-first design:** Each detection includes explicit response actions, not just alert logic. The detection chain is designed to produce the investigation context an analyst needs to act, not just the signal that something happened.
- **False positive discipline:** Each detection includes baseline exclusions and tuning guidance specific to the behaviors most likely to produce false positives in enterprise environments.

5.2 Detection Risk Tiers

The 23 detections are organized into three risk tiers, calibrated against the ARiES actor risk scoring methodology from the Anthropic dataset:

- **Tier 1:** Commodity tier (ARiES 0–40): Detections 1, 5, 12, 15, 17, 22, 23. Maps to the 55–67% prevalence band of behaviors observed across all actor classes. Fires on the most common AI-assisted attacker patterns.
- **Tier 2:** High-risk markers (ARiES 40–70): Detections 2, 4, 6, 8, 10, 16, 18, 20. Techniques 3–5× more common among high-ARiES actors. Indicates post-access operations by a capable threat actor.
- **Tier 3:** GTG-1002 agentic kill chain (ARiES 70–100): Detections 3, 7, 9, 11, 13, 14, 19, 21. Models the highest-risk actor patterns in the dataset. Treat any firing detection in this tier as a potential nation-state or advanced persistent threat operation.

5.3 Selected Detection Deep-Dives

D-03 | Agentic Lateral Movement + Exfiltration | CRITICAL | 24-hr window

Rationale: This detection directly models the GTG-1002 kill chain, the highest-risk actor observed in the Anthropic dataset (ARiES score 100). Lateral movement was the single strongest ARiES predictor, elevating average actor risk by 10.5 points. The full chain, lateral movement via remote services using harvested credentials → web shell deployment → automated bulk exfiltration, distinguishes nation-state and advanced threat actors from the commodity population. The 24-hour correlation window specifically accounts for deliberate slow-burn exfiltration designed to evade volume-threshold alerts.

Detection Logic, Three sequential behavioral markers must all fire within the 24-hour window:

- **Step 1:** Lateral movement via remote services (T1021 + T1078.003): SSH, SMB, or RDP connections from a host that is not a known jump server or admin workstation, using credentials recently created or not previously seen on this host, or sourced from cloud metadata services. Multiple sequential SSH/SMB connections to different internal hosts from the same source within minutes indicates automated movement that cannot be human-paced.

- **Step 2:** Web shell written to a web-accessible directory on any internally reached server (T1505.003): New file creation in webroot directories (/var/www, inetpub, public_html) with script extensions (.php, .aspx, .jsp), written by a process that is not a known web deployment tool, followed by an outbound HTTP POST to that file path from an external IP shortly after creation.

- **Step 3:** Automated bulk data collection and outbound transfer (T1020): Repeated or scripted reads across multiple directories containing documents, keys, configs, or environment files. High-volume outbound data transfer to a single external destination. Archive files created and immediately deleted post-transfer, the staging-and-wipe pattern characteristic of deliberate evidence destruction.

Severity Rationale: Treat any firing of this detection as a potential nation-state or advanced persistent threat operation. Initiate full incident response immediately. Block all outbound connections from involved hosts; enumerate all internal hosts the source has successfully authenticated to during the correlation window; assume all are compromised; revoke all cloud service account tokens and OAuth credentials accessible from the affected environment.

D-14 | Agentic MCP Scaffolding Abuse | CRITICAL | 60-min window

Rationale: This detection targets the behavior that the Anthropic report explicitly states is not yet captured in MITRE ATT&CK: autonomous kill chain orchestration via AI agent scaffolding. GTG-1002 weaponized Claude Code running on a Kali Linux machine, integrating open-source penetration testing tools as MCP servers, effectively turning the AI into an autonomous attack platform. The AI executed commands, made real-time tactical decisions, and chained reconnaissance through exploitation through lateral movement through exfiltration with minimal human input.

The report states that agentic scaffolding, not technique count, is the true differentiator of the highest-risk actors. This detection captures the operational signature of that scaffolding: an AI agent process spawning penetration testing tooling as child processes, with execution patterns that exhibit the machine-speed, low-jitter timing characteristic of AI-driven orchestration rather than human-paced attack operations.

Detection Logic, All four markers within a 60-minute window:

- **Step 1:** Command interpreter spawning known offensive tooling (T1059): Shell or scripting interpreter process spawning recognized penetration testing binaries (nmap, sqlmap, nikto, nuclei, metasploit framework components, impacket scripts) or generic process execution in rapid succession with minimal inter-execution jitter.

- **Step 2:** API-based process creation indicating programmatic orchestration (T1106): Native API calls to process creation functions at a frequency and pattern inconsistent with interactive human use, execution sequences with sub-second inter-process timing, characteristic of agent-orchestrated tool chaining.

- **Step 3:** User execution of tools flagged as malicious by heuristics (T1204.002): Tools executed from non-standard paths without installation artifacts, or binaries that match known offensive framework signatures being executed under processes associated with AI assistant applications.

- **Step 4:** Network communication using standard application protocols (T1071): Outbound connections from the orchestrating process or its children to external targets, particularly using HTTP/HTTPS or DNS with timing patterns consistent with automated reconnaissance or data exfiltration.

6. AI Signal/Detection Authoring

6.1 The Problem: Boiling the Ocean

The traditional detection engineering lifecycle is a manual, labor-intensive grind, often described as "boiling the ocean." It is an exercise in managing infinite complexity across heterogeneous data sources. To write a single high-fidelity detection, an engineer must manually map disparate schemas from dozens of vendors, write and test brittle, regex-heavy rules against inconsistent telemetry, and validate logic against limited, often noisy production data that rarely contains the actual threat pattern being targeted.

The math is unforgiving. Hundreds of ATT&CK techniques multiplied by dozens of log sources produces a matrix no team can staff against, and the size of that matrix is itself the problem. Faced with a backlog that can never be cleared, engineers rationally avoid the expensive cells: the unfamiliar vendor, the deeply nested schema, the source that would require a week of field archaeology before a single where clause could be written. Coverage drifts toward whatever is easiest to author rather than whatever matters most. This is a chilling effect, and its output is not just fewer detections but worse ones.

This "boiling" process creates three primary failure modes:

- **Mapping Friction:** Engineers spend more time translating vendor-specific field names than architecting detection logic. Every new log source triggers a re-mapping exercise, creating a scaling bottleneck that keeps detection coverage perpetually behind attacker innovation. In time, mappings become stale and out of date, causing this problem to manifest in tech debt that is often unnoticed until it’s too late.

- **Testing Blindness:** Because detection engineering rarely happens with the “perfect” labeled data in hand, teams often build rules on hope, writing logic against a mental model of an event that may not match the reality of the raw telemetry.

- **Validation Paralysis:** Validating a rule against live data is reactive and slow, often requiring iterative tuning cycles that bleed resources and delay deployment of critical protections.

None of this work is beyond human capability. Field mapping, schema reading, and fixture construction are all tractable. They're just tedious, and tedium at this scale has a real cost in coverage. It also happens to describe precisely what language models are good at. Reading a nested JSON schema and reconciling it against a target shape is structured pattern work, the model does it in seconds rather than hours, and it does it without the fatigue that makes the fortieth field mapping worse than the first.

LLMs, paired with direct access to raw telemetry streams and vendor documentation, invert the bottleneck. By ingesting technical documentation as the source of truth, an agent can understand the structure of a vendor's telemetry, generate synthetic test fixtures that mirror expected behavior, and

map fields autonomously. Detection engineering shifts from a manual coding task to a model-assisted verification workflow, where the engineer reviews and approves rather than transcribes. The boiling ceases not because the complexity disappears but because the agent absorbs the normalization and testing, and with the marginal cost of a new source no longer prohibitive, the chilling effect lifts. Engineers get to spend their judgment on the part that actually requires it: defining adversarial patterns instead of wrestling with vendor syntax.

6.2 Schema Mapping via Prompts, not Skip Logic

The traditional way to handle field mapping across vendors is a tree of hardcoded conditionals: if the source is Sysmon, look for Image; if it's CrowdStrike, ImageFileName; if it's Okta, actor.alternateId. The tree branches deeper with every new vendor and every schema revision, and it has to be extended by hand each time a source is onboarded.

We replace branching with translation. The signal schema, the user's own or a sensible default, is just a statement of intent: these are the fields I care about, and this is what I want them called. Expressed as a small JSON blob, it becomes a target rather than a set of conditions. Handed to the model alongside the raw events themselves, mapping collapses into a single inference step. The question isn't, "Does this log match one of my known shapes," but, "Given this vendor's field names, what corresponds to the fields I asked for?" The output is another small JSON blob that gets woven directly into the signal's KQL.

This works because authoring is agent driven end to end. Mapping is one of the things the agent does along the way, in the same pass where it reads the vendor documentation, builds a synthetic event, and writes the triggering logic. The user rarely maps a field. They declare a schema or accept the default, and the agent reconciles the vendor's vocabulary to it as a byproduct of building the signal. No separate normalization project, no parser to maintain, no onboarding checklist per source.

It compounds with reuse. Once mapped, a signal carries its own normalization, so any detection composing it inherits the mapping for free.

6.3 When You Can’t Test, Synthesize

Defenders rarely have access to pre-labeled, raw telemetry that displays the precise malicious patterns they intend to catch during the detection authoring phase. Relying on the eventual arrival of this data is an ineffective approach that stalls the creation of essential detection events. To bypass this bottleneck, we reverse the traditional sequence of development.

Instead of starting from data and inferring structure, we start from documentation and infer data. A published event spec, whether Sysmon, CrowdStrike, CloudTrail, or Okta system logs, is itself a reliable ground truth about field names, types, nesting, and typical values. The LLM reads the spec, along with the raw events in the dataset, and constructs a synthetic event: a plausible, well formed instance of the log a real system would emit under the conditions the detection cares about. This is the same instinct a detection engineer follows when building a test fixture, compressed into the

authoring step. The synthetic event becomes the fixture the signal is written and validated against, standing in for real data precisely because real data can't be guaranteed to contain the triggering condition on demand.

From it, a signal query is written: narrow logic testing for a specific event type in a specific vendor. The signal is deliberately atomic. It isn't a detection on its own, it's a building block, validated to confirm it fires when it should and stays quiet on benign lookalikes.

Detections then query signals rather than reimplementing logic, and that's where the efficiency lands. If "process spawned from an Office document" is already validated for a vendor, five detections tracking different follow-on behaviors (beaconing, credential dumping, lateral movement) all draw on the same signal. The logic is tested once, understood once, maintained once, and canonized once. When a field name changes or a new log format ships, there's one place to update instead of a scatter of near duplicates quietly drifting apart. Detections also read as compositions of known good primitives rather than bespoke one-offs, which makes review faster.

6.4 Dealing with Poor Documentation

In the absence of raw, labeled telemetry, which remains notoriously elusive and difficult to replicate within production environments, technical documentation emerges as the primary source of truth. As an example, bespoke internal applications or systems often suffer from chronic documentation gaps. For these assets, organizations must shift toward an autonomous documentation model: deploying agentic LLMs to traverse internal codebases and generate technical artifacts at scale. These agent-derived schemas and logic definitions then function as the formal grammar for security operations, providing the empirical foundation required for high-fidelity signal extraction and detection engineering in the AI era.

The security community should demand robust, high-quality documentation from commercial software vendors. Mirroring the industry's open sharing model for vulnerabilities, log structure documentation has become essential for global community security, particularly within complex microservice architectures, a design paradigm poised for a resurgence as AI simplifies its management.

Synthetic events are only as trustworthy as the docs behind them, so incomplete or stale documentation yields fixtures that miss the same edge cases. Reuse only pays off if the signal library stays organized. An unindexed pile of similar but not identical "suspicious PowerShell" signals defeats the purpose as thoroughly as reusing nothing. The value is in disciplined atomicity and honest bookkeeping about what's been validated against a fixture versus proven against real data.

6.5 Encoding Best Practices in System Prompts

The interesting part of an agentic authoring system isn't the model, it's the instruction layer. Everything we know about writing good Cribl Search KQL is encoded once, in system rules, and applied on every authoring run. The agent doesn't rediscover best practices per task, it inherits them.

The most consequential of those rules is pipeline order, stated as a non-negotiable: dataset → where → extend → project. Filters go against native nested vendor paths first. Only after rows survive that filter does the query extend schema and entity fields, and only then does it project the output shape. Normalization is work performed on a survivor set, not on a firehose. Export is the last step and isn't the model's job at all: the platform appends the export macro on save, and a query that tries to write its own gets rejected.

That division of labor runs deeper than the export line. A large set of fields is declared platform managed and explicitly withheld from the model: signal_id, signal_name, signal_version, event_time, event_key, and the MITRE identifiers. The prompt tells the agent to omit them, and the platform injects canonical values afterward. Deduplication is the clearest case. The event_key expression is derived deterministically from the vendor catalog and verified sample fields, stored in KV, and stamped into the query after generation. The agent is left to do the one thing it's actually best at: reading vendor logs and writing the condition.

Several other constraints are worth naming.

- **Ban the operators that don't exist.** join, mv-expand, parse, array_length, set_difference, pack_array, and friends are listed as forbidden in the prompt and rejected deterministically in validation. Set membership is expressed as values() plus has. Rather than letting the model reach for Azure Kusto habits and fail at runtime, the prompt closes the door up front.

- **Never guess a field name.** The agent gets a verified field list and real sample events, with instructions to use only those paths.

- **Test it, and turn runtime errors into instructions.** When Cribl returns a Kusto error, it's mapped to targeted fix hints and appended to the repair prompt. The model doesn't get told "it broke," it gets told "every stage must start with |," or "prefer todouble(tostring(field)), to_long is Azure-only." Structural problems the platform owns are repaired deterministically and never sent to the model at all.

- **Enforce tiering in language.** Signals detect one atomic technique from one raw dataset. Detections correlate signals over a window and read only the signals sink. The prompt instructs the agent that if the analyst describes a multi-step chain, pick the single strongest indicator, because chains get composed later. Coverage planning reinforces it from the other side: review existing inventory first, reuse what's there, and only author net-new atomic signals.

- **Signal deduplication happens in the detection layer.** Signals are packed with entropic IDs that are used to group duplicates together via the detection KQL.

- **Could enrichment sources help?** Take an inventory of available enrichment, decide if any of these sources could be used to refine signals, reduce noise and amplify true positives.

The theme across all of it is that the agent is given a narrow, well-lit job and the platform owns everything that can be decided without inference. Best practices live in the prompt, correctness lives in validation, and the expensive work happens last.

6.6 Signal Maintenance

Signal queries are the layer that rots. Vendors rename fields, add products, and change log formats. Maintaining dozens of vendor-specific queries per source is exactly the burden that ingest-time schema mapping imposes, unless it is automated.

The APEX model packages signal maintenance as an AI workflow built on the Cribl Apps platform. The workflow covers four operational areas:

- **Diagnose:** Drift diagnosis: on a silent-signal flag, the skill samples recent raw events from the affected dataset, compares observed field names and shapes against the fields the signal query references, and identifies the specific mismatch.

- **Repair:** Repair proposal: the skill drafts a corrected query, validates it against live data through MCP, confirming it parses, returns plausible hits, and emits schema-conformant signals, and presents a diff for human approval. Approved changes are committed to the signal library with a version bump.

- **Coverage:** Coverage management: the skill maintains the technique-to-source coverage map against the APEX 23-chain library, identifies gaps, and drafts candidate signal queries for new sources, seeded from existing signals for the same technique in other sources.

The AI proposes, humans approve. Query changes alter what the organization can detect. The approval gate is not optional. Packaged as a single installable App, the workflow is the difference between a methodology document and a deployable product.

7. Conclusion: A Detection Program That Survives the AI Era

IOCs are not going away. They remain useful for blocking known-bad in volume, for enriching alerts with context, and for retroactive hunting once a campaign is disclosed. What is going away is their role as the primary detection surface. Generative AI has commoditized the production of disposable indicators, and no IOC feed can keep pace with a model that mints new infrastructure and unique binaries on demand.

The APEX model is not a product claim or a vendor roadmap entry. It is the formalization of a detection philosophy that the security operations community has been building toward for over a decade, and in many ways over-corrected for by leaning too heavily on data reserialization. Now made urgent by the empirical reality documented in the Anthropic Frontier Red Team dataset and emerging recent incidents. Eight hundred and thirty-two threat actors. Thirteen thousand eight hundred and seventy-three TTP observations. Twelve months of AI-enabled attacker behavior. The data is clear: the shift is already underway, the highest-risk actors are already operating as autonomous agents, and the detection programs that have not made the architectural pivot are already behind.

The defensive answer has three components, and APEX addresses all three:

- **Measure correctly:** Codify TTP coverage in ATT&CK terms, and measure it not as a rule count but as a behavioral chain coverage map. Ask not 'do we have a rule for T1021?' but 'do we detect T1021 chained with T1505.003 and T1020 within a 24-hour window?' Those are different questions with different answers.

- **Instrument correctly:** Instrument behaviors, not artifacts. The data pipeline must deliver the telemetry that behavioral detection requires, enriched, normalized, timely, and complete. If the pipeline cannot provide full process telemetry with command-line arguments, parent-child process relationships, network flow context, and identity events in correlation, the behavioral detection layer cannot function.

- **Learn continuously:** Close the loop. Behavioral detection that does not feed back into the data pipeline that feeds it will not improve. The learning loop is not a feature, it is the structural requirement that distinguishes a detection program that gets better over time from one that plateaus at its initial deployment quality.

The detection programs that thrive over the next several years will be the ones that stopped trying to win at the bottom of the Pyramid of Pain and redesigned for the top. APEX is the blueprint for that redesign.

> *The technology to deliver actual behavioral detection, not the vendor promise of it, but the operational reality, exists today. The architecture is defined. The detection library is built.*

8. References

[1] Anthropic Frontier Red Team. AI-enabled cyber threats: Insights from 832 threat actors. https://www.anthropic.com/news/AI-enabled-cyber-threats-mitre-attack. June 2026.

[2] Anthropic Frontier Red Team. Mapping AI-enabled cyber threats: Insights from the LLM ATT&CK Navigator. https://www.anthropic.com/research/attack-navigator. June 3, 2026.

[3] Wang, F., Li, Y., Shao, Z., Yu, C., Fu, Y., An, Z., Xu, Y., & Cheng, X. ARIES: Relation Assessment and Model Recommendation for Deep Time Series Forecasting. https://arxiv.org/pdf/2509.06060. arXiv:2509.06060, 2025.

[4] Bianco, D. The Pyramid of Pain. Enterprise Detection & Response blog, 2013.

[5] MITRE Corporation. MITRE ATT&CK Framework. https://attack.mitre.org.

