From Prediction to Attribution: Integrating Unsupervised Learning for Enhanced SOAR Triage

# Introduction

Security Operations Centers (SOCs) handle a constant flood of alerts. Most tools that feed a SOAR (Security Orchestration, Automation, and Response) pipeline begin with binary predictions: a URL, file, or event is flagged as malicious or benign. That verdict is helpful for filtering obvious noise, but it does not tell analysts what kind of adversary they are facing, why the activity is happening, or how they should tailor their subsequent actions. In practice, a binary label is often only the first step of triage, not the end result. As adversaries diversify their tactics, techniques, and procedures (TTPs), SOCs need context that turns detection into direction: the who, why, and how behind an alert. This paper presents a pragmatic approach to adding that missing layer by chaining a supervised classifier with an unsupervised clustering stage that performs lightweight attribution. The result is a SOAR‑friendly workflow that preserves the speed of classification while surfacing actor‑like groupings to shape playbooks and communication (Sommer & Paxson, 2010).

# Methodology

## Feature Engineering

Our pipeline focuses on URL‑centric phishing indicators because they are inexpensive to compute, widely studied, and intuitive for analysts. The synthetic dataset encodes features commonly associated with deceptive web campaigns: presence of an IP address in place of a hostname, unusually long URLs, excessive subdomains, prefix–suffix patterns, link‑shortening services, abnormal SSL states, and hints from anchor tags and form handling. These map to well‑documented attacker behaviors. Crimeware kits often rely on disposable infrastructure and link obfuscation to evade reputation checks. State‑aligned operations tend to exhibit cleaner infrastructure but may blend in using look‑alike domains and staged redirects. Ideologically motivated groups sometimes favor broader, noisier campaigns with simpler obfuscation. The point of the synthetic design is not perfect realism; it is to create separable behavioral contours that a clustering algorithm can discover so the second stage can assign an alert to a stable, human‑meaningful category. As behavior‑rule research shows, clear behavior specifications materially affect detection and response quality (Mitchell & Chen, 2014).

Two practical choices keep the features operational. First, we avoid deep content inspection or heavy network context that would slow interactive use. Second, we normalize and encode categorical signals so that distance‑based learners can treat them coherently. Both choices simplify model behavior and make explanations easier to communicate to analysts who must justify decisions under time pressure.

## Algorithm Selection

We adopt a two‑model design. The supervised stage is a standard classifier trained to recognize malicious URLs from benign ones. The unsupervised stage activates only for records deemed malicious. For attribution, we select K‑means clustering. Three properties make K‑means attractive in a SOC context: (1) speed—centroid updates and point assignments are computationally light; (2) interpretability—clusters are summarized by centroids, making it straightforward to describe how features influence membership; and (3) operational stability—given a fixed number of clusters, the outputs are consistent enough to wire into playbooks. While density‑based methods such as DBSCAN and hierarchical linkages can discover irregular shapes and variable densities, they introduce sensitivity to parameters and can yield clusters that appear or disappear as the data mix changes. In triage, stability is often more valuable than perfect geometric fidelity. As Jain emphasizes, K‑means persists in real systems because it balances simplicity, speed, and useful approximations of structure (Jain, 2010).

We purposely set a small number of clusters that align with meaningful actor themes—Organized Cybercrime, State‑Sponsored, and Hacktivist. This prior keeps the output legible to the SOC and the mapping from cluster ID to profile stable across retrains. If future data suggest different natural groupings, the mapping can be revised without rewriting the overall architecture.

# Implementation

The application is implemented in Python using Streamlit for a simple analyst UI and PyCaret/scikit‑learn for modeling. At initialization, the app loads two persisted artifacts: a classification pipeline and a clustering pipeline. The sidebar collects feature inputs that mirror the training schema (for example, URL length, SSL state, presence of an IP address, and whether a link shortener is used). On submission, the app performs the following sequence: (1) the features are sent to the classifier to obtain a malicious/benign verdict and a confidence score; (2) if—and only if—the verdict is malicious, the same feature vector is passed to the clustering pipeline to obtain a cluster label; and (3) the numeric cluster is mapped to a human‑readable actor profile via a table learned once from the training data (majority label per cluster) and then curated by the analyst team. The UI shows the suspected actor, a short narrative describing typical motives and tradecraft, and a risk‑contribution chart that links the UI sliders to the classifier’s decision.

Two engineering choices keep the UI responsive. First, Streamlit caches model artifacts so they are loaded once per session. Second, the clustering output is normalized to an integer before lookup to prevent version‑specific formatting quirks from breaking the mapping. If attribution fails for any reason (missing artifact, schema mismatch), the app gracefully degrades to classification‑only mode so analysts are never blocked. Finally, the app integrates a prescriptive overlay: once a case is flagged as malicious, a GenAI provider drafts recommended actions and a communication template aligned with the suspected actor. These drafts are clearly labeled as guidance to be reviewed by a human—consistent with warnings about over‑reliance on machine‑generated outputs (Sommer & Paxson, 2010).

# Results & Discussion

The combined pipeline improves triage ergonomics. In testing with the synthetic dataset, the classifier separates benign from malicious inputs, and the clustering stage splits malicious cases into behaviorally coherent bins that match the defined actor themes. The immediate operational gain is in playbook branching: rather than a single 'malicious → blocklist' path, the SOAR can route cases to actor‑specific responses. Analysts report faster decision‑making because the UI surfaces a concise explanation (risk features) and a plausible attribution, reducing the time spent forming a hypothesis about the nature of the campaign.

There are limitations. Clustering is unsupervised; its labels are interpretations of structure, not ground truth. Misattribution is possible, especially when attackers change tooling or features drift. The mitigation is process‑level: treat attribution as a probabilistic advisory signal, require human confirmation before high‑impact actions, and monitor cluster cohesion over time. Retraining on fresh data and periodically revisiting the mapping table help keep the system aligned with reality (Jain, 2010; Sommer & Paxson, 2010). A second limitation is data realism. Our prototype uses synthetic distributions tuned to create separable behaviors. This is acceptable for designing and validating the pipeline, but production deployment should incorporate historical alerts, campaign indicators, and telemetry from email gateways, DNS, and proxies. Behavior‑rule work underscores the value of specifying and validating adversary behaviors against real systems when safety or business impact is high (Mitchell & Chen, 2014).

Despite these caveats, the benefits are concrete: fewer generic alerts, clearer routing decisions, and prescriptive guidance that matches the suspected adversary. The two‑stage design also creates a seam for enrichment, such as linking clusters to MITRE ATT&CK tactics or adding weak signals from external intelligence feeds. Because K‑means is fast, the added attribution step imposes no noticeable latency in the analyst experience, preserving the responsiveness that SOC workflows demand.

# Conclusion

Binary detection tells a SOC that something is wrong. Attribution explains the type of error and what to do next. By chaining a supervised classifier with a simple, interpretable clustering stage, this project turns a phishing‑URL verdict into an attributed incident with actor‑appropriate playbooks and communication. The method is intentionally modest—compact features, fast models, straightforward UI—but that is its strength: it fits how SOCs work. With real telemetry, periodic retraining, and human‑in‑the‑loop confirmation, the same architecture can move from classroom prototype to production‑grade triage. Most importantly, it shows how SOAR can evolve from prediction to actionable context without sacrificing speed or transparency.

# References

Jain, A. K. (2010). Data clustering: 50 years beyond K‑means. Pattern Recognition Letters, 31(8), 651–666.

Mitchell, R., & Chen, I.-R. (2014). Behavior rule specification‑based intrusion detection for safety‑critical medical cyber‑physical systems. 1-14

Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In 2010 IEEE Symposium on Security and Privacy (pp. 305–316).