Informatics, Electrical and Electronics Engineering (Infotron)
Volume 6, Number 1, (2026) pp. 28-43
ISSN 2798-0197
Sinta 4 Accreditation – No SK. 177/ E/ KPT/ 2024
http://riset.unisma.ac.id/index.php/infotron/article/view/24923
DOI: 10.33474/infotron.v6i1.24923

Interpretable Attack-Chain Stage Detection from AWS CloudTrail Event Sequences via Linear Models and HMM Smoothing

Jingwen Bai a[1], Siyu Chen b[2], Daren Zheng c[3], Meng-Ju Kuo d[4]*
[a]Data Science, Columbia University, NY, USA
[b]Information Management, University of Illinois Urbana-Champaign, IL, USA
[c]Information Technology, Carnegie Mellon University, PA, USA
[d]Department of Electrical and Computer Engineering, Carnegie Mellon University, PA, USA
[1]jwbai1118@outlook.com, [2]chensy010227@gmail.com, [3]darenzheng951@gmail.com, [4]mengju.kuo0688@outlook.com*

**corresponding author*

Article History:

Received 19 January 2026
Reviewed 21 January 2026
Revised 21 March 2026
Accepted 8 May 2026

A B S T R A CT

Cloud audit logs such as AWS CloudTrail are a primary source of evidence for both incident response and compliance auditing, yet operational alerting still struggles with two coupled requirements: (i) recognizing multi-step cloud attacks from sequences of API events, and (ii) producing explanations that analysts and auditors can directly verify. This paper presents a reproducible and interpretable baseline pipeline for attack-chain stage detection from CloudTrail event sequences. Using the public Stratus Red Team AWS detonation logs (v2.23.2), a small labeled dataset containing 310 CloudTrail events across 35 AWS techniques and 10 ATT&CK tactics, we study three tasks: event-level tactic classification, windowed sequence classification, and attack-chain segmentation. Each CloudTrail event is encoded into a compact, schema-aware token set derived from native fields such as service, operation, principal type, management flags, and request/response key signatures, and transparent linear classifiers (TF-IDF + LinearSVC / logistic regression) are used to predict ATT&CK tactics. To recover a more coherent stage timeline, we smooth per-event predictions with a Hidden Markov Model (HMM) and Viterbi decoding using synthetic multi-stage chains assembled from the published technique traces. We further generate coefficient-based evidence reports that link predicted stages to high-weight events and feature tokens. On event-level tactic recognition, LinearSVC achieved 0.961 accuracy and 0.866 macro-F1. On synthetic multi-stage chains, LR + HMM improved boundary-F1 from 0.800 to 0.892, while accuracy and macro-F1 increased slightly from 0.840 to 0.848 and from 0.873 to 0.877, respectively. These results should be interpreted as a controlled public baseline rather than a production-ready estimate, but they show that simple and transparent models can provide useful stage-aware summaries and auditable evidence for cloud audit-log analysis.

Keywords: CloudTrail; cloud audit logs; attack chain; MITRE ATT&CK; sequence modeling

1. Introduction

Cloud infrastructure centralizes identity, networking, storage, and compute control into API-driven management planes. In AWS, most administrative actions and many data-plane operations are mediated by API calls that can be recorded by AWS CloudTrail, which makes CloudTrail a natural backbone for security monitoring and audit evidence [1], [2]. For regulated organizations, CloudTrail is also routinely referenced in internal controls to demonstrate accountability for privileged actions, configuration changes, and access to sensitive resources [7].

CloudTrail is particularly valuable because its event schema is explicit. Each record contains eventSource (the service namespace), eventName (the API operation), userIdentity attributes (principal

Informatics, Electrical and Electronics Engineering (Infotron)
Volume 6, Number 1, (2026) pp. 28-43
ISSN 2798-0197

type and assumed role patterns), and request/response fields that reveal what was attempted and what objects were affected. CloudTrail also distinguishes management events from data events via eventCategory and managementEvent flags [1], [2]. In practice, these schema elements enable a monitoring program to answer questions that are central to both incident response and compliance: Who performed a sensitive action? From which identity and session? Which control-plane API was invoked? Did it succeed or fail?

At the same time, cloud-native threats increasingly manifest as sequences of legitimate API operations rather than easily signatured malware artifacts. An attacker can obtain access keys, enumerate resources, change IAM policies, disable logging, execute commands through managed services, and exfiltrate data while only emitting syntactically valid API calls. This shifts detection from binary event matching to contextual reasoning over sequences. A single DescribeInstances call is rarely malicious on its own; a DescribeInstances call followed by SSM SendCommand and CloudTrail StopLogging forms a coherent chain that strongly suggests malicious intent. Effective cloud monitoring therefore requires attack-chain reasoning, not just per-event alerting.

Attack-chain reasoning is often operationally represented as stages. MITRE ATT&CK provides a widely used taxonomy of adversary tactics and techniques, including a cloud matrix that captures how adversaries interact with cloud control planes [6]. Mapping observed CloudTrail events to ATT&CK tactics provides two benefits. First, it yields a human-comprehensible stage timeline (e.g., credential-access → discovery → execution). Second, it gives a standard language for communicating findings in incident reports, security reviews, and audit narratives. In many organizations, incident postmortems and governance documentation explicitly reference ATT&CK to justify detection coverage and response decisions.

Existing cloud monitoring practice often relies on hand-written detections: indicator-based rules (e.g., known malicious IPs), API-level heuristics, or portable rule formats such as Sigma [20]. Rules can be precise for known behaviors but are expensive to maintain as AWS services evolve, and they typically fire on a single event with limited context. To reduce integration friction across tools, organizations increasingly normalize events into standard schemas such as the Open Cybersecurity Schema Framework (OCSF) [19]. Normalization helps, but it does not solve the higher-level problem: correlating events into multi-step attacks and producing explanations that are acceptable to both analysts and auditors.

Machine-learning methods for log analysis and cyber detection have been studied extensively [12]-[18], [21]-[30]. Early work mined logs for anomaly signatures or clusters of related messages [14]. Modern systems use log parsing (e.g., Drain) to extract structured templates, then apply sequence models to detect anomalies [13]. Deep learning approaches such as DeepLog learn event-sequence patterns and can flag deviations [12], while transformer-based models (e.g., LogBERT) leverage contextual embeddings for representation learning [15], [18]. These approaches demonstrate that sequential structure in logs is informative. However, many deep models introduce explainability challenges in security operations, where alert validity must be defensible and model behavior must be governable.

Explainable AI has therefore become a key requirement for security analytics. Post-hoc explanation methods such as LIME and SHAP can approximate feature attributions for arbitrary models [16], [17]. In security settings, explanations are most useful when they can be traced back to raw evidence without ambiguity. Linear models are attractive because their attributions are intrinsic: coefficients directly quantify how each feature contributes to a prediction. This provides a tight link between the alert and the underlying audit log fields.

Sequence labeling is another classical area with strong interpretability properties. Hidden Markov Models (HMMs) represent sequences with a small number of latent states and explicit transition probabilities [9]. Conditional Random Fields (CRFs) generalize this idea and can incorporate rich features while preserving a probabilistic interpretation [11]. In cloud audit logs, the explicit state

DOI: 10.33474/infotron.v6i1.24923

Informatics, Electrical and Electronics Engineering (Infotron)
Volume 6, Number 1, (2026) pp. 28-43
ISSN 2798-0197

sequence is especially valuable: it corresponds to an attack-stage timeline that analysts can validate. We therefore use an HMM as a smoothing layer on top of transparent per-event predictions.

This paper targets interpretable, stage-aware analytics for CloudTrail event streams. We focus on three practical recognition problems: (i) classifying individual CloudTrail events into ATT&CK tactics, (ii) classifying short windows into tactics to capture local ordering, and (iii) segmenting longer streams into stable stage timelines (attack chains). Our goal is not to claim a fundamentally new learning algorithm, but to establish a transparent baseline in which every stage prediction can be traced back to native CloudTrail fields and a small, auditable set of evidence tokens.

A central barrier for research [31-40] on cloud audit-log attack chains is the lack of public, labeled datasets with realistic CloudTrail semantics. We address this by evaluating on the Stratus Red Team AWS detonation logs, which publish CloudTrail events generated by executing documented adversary techniques against AWS [3]. The logs are generated with the Grimoire project and anonymized with LogLicker to support sharing without leaking sensitive identifiers [4], [5]. Because the public dataset is small (310 events across 35 techniques), we position it as a controlled benchmark for reproducible baseline evaluation rather than as a full proxy for operational cloud traffic.

The contribution of this paper is therefore primarily integrative and operational. First, we define a schema-aware feature representation that stays close to the native CloudTrail fields (eventSource, eventName, userIdentity.type, and request/response key sets) and remains interpretable after anonymization. Second, we combine transparent per-event classifiers with HMM-based sequence smoothing to produce contiguous ATT&CK stage timelines and explicit segment boundaries. Third, we operationalize interpretability by translating linear model weights into evidence snippets that can be copied into tickets and audits. All experimental results reported in this paper are empirically measured on the specified public dataset and should be read as baseline results under controlled conditions.

From a governance standpoint, the same CloudTrail evidence is routinely repurposed across monitoring, incident response, and audit. For example, an investigation may start from a SOC alert, but the closure of the incident often requires documenting which privileged actions occurred, who executed them, which resources were affected, and whether preventive controls were bypassed. This motivates explanations that are not only locally faithful to the classifier but also exportable as structured "evidence rows" (API call, principal type, key request schema tokens, and time). Our report generator is designed for that workflow: it summarizes each predicted stage in plain language, enumerates the highest-impact CloudTrail events, and attaches the ATT&CK mapping as well as a small set of feature-level rationales. These properties are aligned with common security-control families (audit logging, least privilege, and change management) and reduce the manual effort needed to translate raw event JSON into an auditable narrative [7], [8]. We also follow normalization conventions used in community detection content and schemas; in particular, the evidence fields we emit correspond closely to the primitives used by Sigma detections and cloud logging schemas [20], [21].

2. Method

This section presents the complete pipeline for attack-stage recognition and explainable reporting from CloudTrail sequences. The pipeline consumes raw CloudTrail records, normalizes the schema, performs feature and sequence encoding, predicts ATT&CK tactics with transparent linear models, smooths predictions with an HMM to recover a coherent stage timeline, and finally produces a structured alert report (Fig. 1).

Problem formulation. Let E = (e1, e2, ..., eT) denote a time-ordered sequence of CloudTrail events. Each event ei is a JSON record that includes eventSource, eventName, userIdentity information, and request/response metadata [1]. Our primary objective is to infer a sequence of latent attack stages S = (s1, s2, ..., sT), where each st is an ATT&CK tactic label (e.g., execution, credential-access). We also produce a segmentation of the stream into contiguous stage segments and generate an explanation for each segment. A secondary objective is technique identification, where a short sequence is labeled with a specific Stratus technique identifier (e.g., aws.execution.ssm-send-command).

DOI: 10.33474/infotron.v6i1.24923

Informatics, Electrical and Electronics Engineering (Infotron)
Volume 6, Number 1, (2026) pp. 28-43
ISSN 2798-0197

Dataset and labeling. We use the Stratus Red Team detonation logs v2.23.2 [3]. Each JSON file corresponds to one AWS technique and contains the CloudTrail events observed during a controlled detonation of that technique. The technique identifier follows the pattern aws.<tactic>.<technique-name>, which we use to derive a tactic label for every event in that detonation. Table I summarizes the dataset statistics and Table II gives the per-tactic event and technique counts. The dataset contains 35 techniques spanning 10 tactics and 310 CloudTrail events. This scale is sufficient for a transparent baseline study, but it is small for machine-learning evaluation and should not be treated as a comprehensive sample of real-world cloud activity. The distribution is imbalanced: execution and credential-access dominate the event volume (Fig. 2), while initial-access and privilege-escalation are represented by a single event each (Table II).

Data ingestion and preprocessing. We parse every detonation JSON file and treat it as one canonical technique trace. Within each trace, events are already time-ordered by eventTime. We discard identifying strings that are not stable under anonymization (e.g., ARNs and account IDs) and focus on stable semantic and structural fields. This is consistent with real enterprise sharing and governance, where audit logs are often pseudonymized before analysis or before transfer to a central SOC.

Parsing and normalization. CloudTrail records contain nested fields and optional keys. We extract a compact set of attributes that are stable across AWS services and CloudTrail versions. We treat absent fields explicitly rather than imputing values. For example, an absent errorCode is encoded as err=None, which is distinguishable from a present errorCode and is itself meaningful because many adversary behaviors involve repeated failures before success.

Schema-aware feature encoding. For each event, we build an interpretable feature string composed of tokens derived from core fields and key sets. Table III lists the feature tokens and their sources. Core tokens include src=<eventSource>, name=<eventName>, userType=<userIdentity.type>, readOnly, managementEvent, eventCategory, eventType, region, and an errorCode token. To capture request/response structure while staying robust to anonymization, we add tokens for the keys present in requestParameters and responseElements (reqKey=..., respKey=...). We also include resource type hints (resType=...) and selected additionalEventData keys (aedKey=...) when present. This representation preserves interpretability: every token is directly traceable to a CloudTrail field or schema element.

Example of feature text. Consider an SSM command execution event with eventSource=ssm.amazonaws.com and eventName=SendCommand. The resulting feature text includes tokens such as src=ssm.amazonaws.com, name=SendCommand, userType=IAMUser, mgmt=True, and request schema hints like reqKey=instanceIds and reqKey=parameters (when present). This makes the model output auditable: the explanation references the same field names that an analyst sees in the raw record.

Vectorization. We convert feature strings into sparse vectors using TF-IDF. For a token t in event i, TF-IDF is tf(t,i) * log(N/df(t)), where N is the number of events in the corpus and df(t) is the document frequency. TF-IDF downweights ubiquitous tokens (e.g., generic AWS strings) and emphasizes discriminative tokens (e.g., service-operation pairs and rare request keys). Because the vocabulary is derived directly from CloudTrail schema tokens, the resulting vector space remains interpretable.

Event-level tactic classifier. Given the TF-IDF vectors, we train linear models to predict tactics. We evaluate multinomial logistic regression (LR) and linear SVM (LinearSVC). LR models class probabilities as $p(y=k|x)=\text{softmax}(Wx+b)$, which supports probabilistic emissions for HMM smoothing. LinearSVC optimizes a max-margin objective and often performs well in high-dimensional sparse spaces [10]. To mitigate class imbalance in tactic recognition, we apply balanced class weights for LR and LinearSVC and report macro-F1 in addition to accuracy and weighted-F1. We do not apply random oversampling because classes with only one labeled event would simply duplicate the same example and would likely overstate confidence; on larger datasets, resampling or class-balanced learning could be explored. We also evaluate multinomial naive Bayes (NB) with count features and a majority baseline. For all event-level results, we perform 5-fold stratified cross-validation over the 310 events.

DOI: 10.33474/infotron.v6i1.24923

Informatics, Electrical and Electronics Engineering (Infotron)
Volume 6, Number 1, (2026) pp. 28-43
ISSN 2798-0197

Windowed sequence classifier. Some tactics are better recognized from short sequences rather than isolated events. We therefore construct sliding windows of length four events from each technique trace (when available) and classify each window into a tactic. Windows are encoded as token sequences of the form <eventSource>:<eventName>. We represent these windows with TF-IDF n-grams (1-2) over the token stream and evaluate LinearSVC and one-vs-rest logistic regression. This window setting captures local ordering patterns while remaining interpretable, since the n-grams correspond to short API call subsequences that can be directly inspected.

Technique identification with augmentation. The public dataset provides one canonical detonation trace per technique, so direct train/test evaluation at the technique level is underdetermined. To obtain a controlled proxy evaluation that preserves dataset consistency, we generate multiple augmented sequences per technique by stochastically dropping events (drop probability 0.2) while preserving order and optionally duplicating one event (probability 0.25). The intuition is to approximate minor variability across repeated detonations, such as missing ancillary events, logging differences, or API retries, without introducing tokens that are not present in the original trace. This augmentation is used only for the technique-identification proxy task and is not intended to model the full variability of production CloudTrail streams. We generate 25 augmented sequences per technique and perform a stratified 80/20 train/test split.

Attack-chain smoothing with an HMM. Linear per-event classifiers can produce locally inconsistent stage labels (e.g., a single misclassified event within an otherwise coherent stage). We address this by modeling the latent stage sequence S as a first-order Hidden Markov Model (HMM) [9]. The HMM has a start distribution p(s1), a transition matrix p(st|st-1), and emission probabilities p(xt|st). We derive emissions from LR class probabilities for each event. We then apply Viterbi decoding to compute the most likely stage path. Viterbi uses dynamic programming over time: dp[t,k] = max_j dp[t-1,j] + log p(k|j) + log p(x_t|k). The resulting path yields a smoothed stage timeline and stage boundaries.

Transition estimation. We estimate p(st|st-1) and p(s1) from labeled sequences. Because transition counts can be sparse, we apply add-one smoothing. The estimated transition matrix is directly interpretable: it reveals which tactic transitions are frequent in the training chains and therefore favored by the decoder.

Synthetic chain construction for training and evaluation. To evaluate segmentation end-to-end, we assemble synthetic multi-stage chains by concatenating complete technique traces. We sample 4-6 tactics per chain without replacement from a fixed tactic order (initial-access, execution, persistence, privilege-escalation, credential-access, discovery, lateral-movement, exfiltration, impact, defense-evasion) and then sample one technique trace per chosen tactic. We create 120 chains to estimate the HMM transition and start distributions and 60 chains for testing. This construction preserves the within-technique event order observed in the published detonation traces and provides explicit ground-truth boundaries for studying sequence smoothing when no public mixed benign/malicious CloudTrail corpus with tactic labels is available. At the same time, these synthetic chains are only a controlled approximation of attack progression and do not fully capture production settings with benign background events, time gaps, or concurrent activity. Each chain provides ground-truth per-event tactic labels and segment boundaries at the start of each technique trace.

Explainable alert report generation. For each predicted stage segment, the report contains: (i) the stage name (ATT&CK tactic), (ii) a timeline range (event indices or timestamps), (iii) top evidence events ranked by classifier confidence margin, and (iv) top contributing feature tokens ranked by the model's coefficients for that stage. For LR, the log-odds for class k is w_k^T x + b_k, so each token contributes additively via its coefficient and TF-IDF value. This enables faithful explanations: the report lists the same tokens that mathematically drove the prediction. Fig. 7 illustrates a template of top features for the execution tactic. Operational report format. The pipeline's final artifact is a structured alert record that can be stored alongside raw CloudTrail evidence. For each stage segment, the report includes a stable identifier (tactic name), start and end indices (and eventTime if available), a compact

DOI: 10.33474/infotron.v6i1.24923

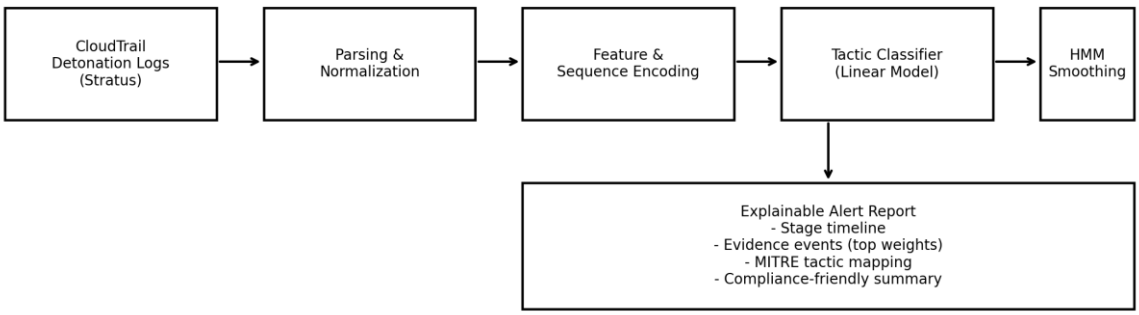
Informatics, Electrical and Electronics Engineering (Infotron)
Volume 6, Number 1, (2026) pp. 28-43
ISSN 2798-0197

evidence set, and an explanation set. The evidence set consists of the top-N events in the segment ranked by the classifier margin (difference between the best and second-best class), which prioritizes records that are both strongly indicative and least ambiguous. The explanation set consists of the top-M feature tokens for the predicted tactic (global coefficients) and the top-M tokens present in the evidence events (local presence). In deployment, these fields map cleanly to incident-ticketing and SOAR payloads: the evidence events can be attached verbatim as JSON, while the token lists provide a short human-readable summary. Because the report is derived deterministically from the model and the input events, it is reproducible and supports after-the-fact auditing. In this paper, the explainability assessment is limited to intrinsic faithfulness of the linear coefficients and qualitative case-study inspection; a quantitative analyst study is left for future work.

The system also renders evidence events back into CloudTrail fields (service, operation, errorCode, request keys) so analysts can validate the alert directly against raw logs.

Evaluation metrics. For classification we report accuracy, macro-F1 (unweighted average over classes), and weighted-F1 (weighted by support). For attack-chain segmentation we additionally report boundary-F1. Boundaries are defined as the start indices of stage segments (excluding the first segment). A predicted boundary is considered correct if it falls within ±1 event of a true boundary; we compute precision, recall, and F1 over boundaries and average across test chains. All results are computed from empirical evaluations on the dataset described above with a fixed random seed (42); all numbers are empirically measured on the specified public dataset and are generated by our evaluation scripts.Model configuration and reproducibility. Across all experiments, we fixed the random seed at 42 and used deterministic data splits (StratifiedKFold for cross-validation and stratified 80/20 splits for train/test). For TF-IDF vectorization, we used the default L2-normalized TF-IDF with min_df=1. For event-level LR, we used multinomial logistic regression with max_iter=2000 and balanced class weights; for technique and window experiments we used one-vs-rest LR with solver='saga' and max_iter=2000 to ensure stable convergence in the multi-class setting. LinearSVC was trained with the default hinge loss and, when applicable, balanced class weights. All tables and figures in this paper are generated by the accompanying scripts that download the specified public dataset (Stratus Red Team v2.23.2) and execute the full evaluation end-to-end.

Mapping stages to reports and controls. The alert report is emitted as a structured object (e.g., JSON) that contains a stage timeline and evidence fields. Each stage segment includes the ATT&CK tactic label [6], the supporting CloudTrail event identifiers (eventTime, eventSource, eventName), and the top explanatory tokens. In practice, this structure supports multiple downstream uses: SOC triage (linking to raw events in a SIEM), incident response (timeline reconstruction), and compliance (attaching evidence to a control such as logging integrity or privileged access monitoring). Because all evidence is drawn from CloudTrail fields, an auditor can independently validate the report by re-running log queries and checking the same fields.



DOI: 10.33474/infotron.v6i1.24923

Informatics, Electrical and Electronics Engineering (Infotron)
Volume 6, Number 1, (2026) pp. 28-43
ISSN 2798-0197

Fig. 1. End-to-end pipeline for interpretable CloudTrail attack-chain detection and alert reporting.

Table 1. Dataset overview (Stratus Red Team detonation logs).

Item	Value
Dataset version	Stratus Red Team detonation logs v2.23.2
Techniques (AWS)	35
Tactics	10
CloudTrail events	310
Unique (eventSource:eventName) tokens	54
Sequence length (min/median/mean/max)	1/2/8.86/56

Table 2. Per-tactic distribution of techniques and events.

tactic	techniques	events	unique_tokens	avg_events_per_technique
execution	4	123	11	30.75
credential-access	5	115	12	23
discovery	1	19	2	19
persistence	10	17	15	1.7
impact	1	15	6	15
defense-evasion	6	8	7	1.33
lateral-movement	2	7	3	3.5
exfiltration	4	4	4	1
initial-access	1	1	1	1
privilege-escalation	1	1	1	1

Table 3. Interpretable feature tokens derived from CloudTrail JSON fields.

Feature token	Source field	Description	Used
src	eventSource	API service namespace (e.g., ssm.amazonaws.com)	Yes
name	eventName	API operation name (e.g., SendCommand)	Yes
etype	eventType	CloudTrail record type (e.g., AwsApiCall)	Yes
ecat	eventCategory	Management or Data event	Yes
region	awsRegion	AWS region string	Yes
userType	userIdentity.type	Principal type (IAMUser, AssumedRole, ...)	Yes
readOnly	readOnly	Read-only flag	Yes
mgmt	managementEvent	Management event flag	Yes
err	errorCode	AWS error code (if any)	Optional
reqKey	requestParameters keys	Schema keys of requestParameters	Optional
respKey	responseElements keys	Schema keys of responseElements	Optional
resType	resources[].type	Resource type strings	Optional
aedKey	additionalEventData keys	Selected flags (e.g., MFAUsed)	Optional

DOI: 10.33474/infotron.v6i1.24923

Informatics, Electrical and Electronics Engineering (Infotron)
Volume 6, Number 1, (2026) pp. 28-43
ISSN 2798-0197

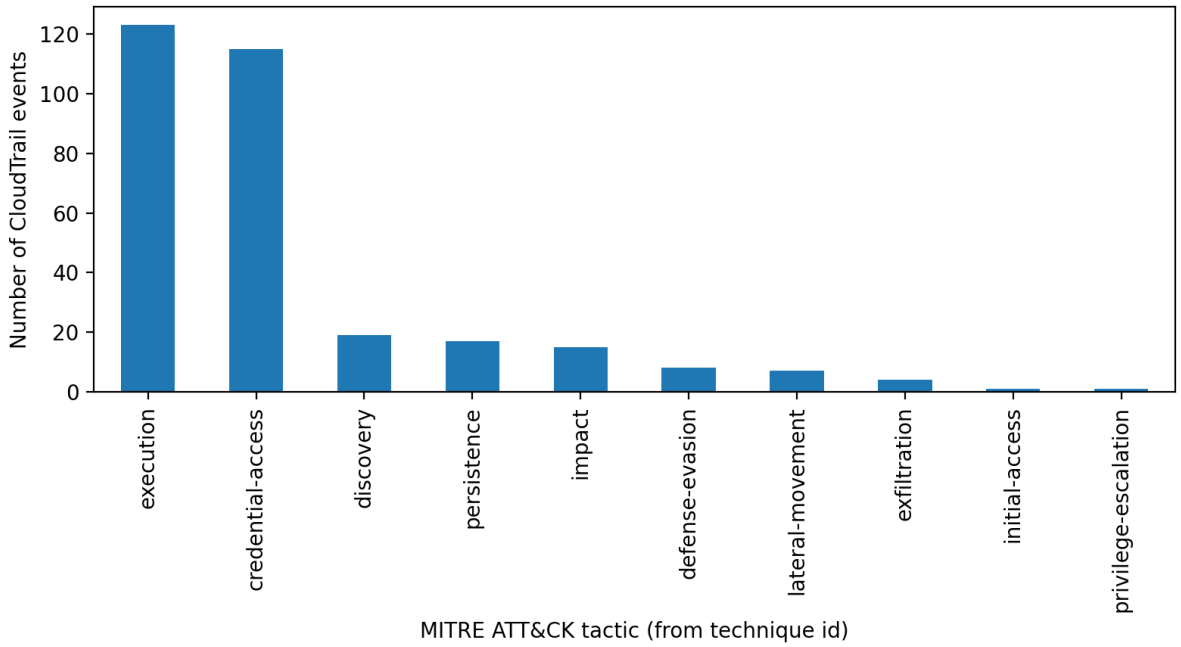


Fig. 2. CloudTrail event counts per ATT&CK tactic in the dataset.

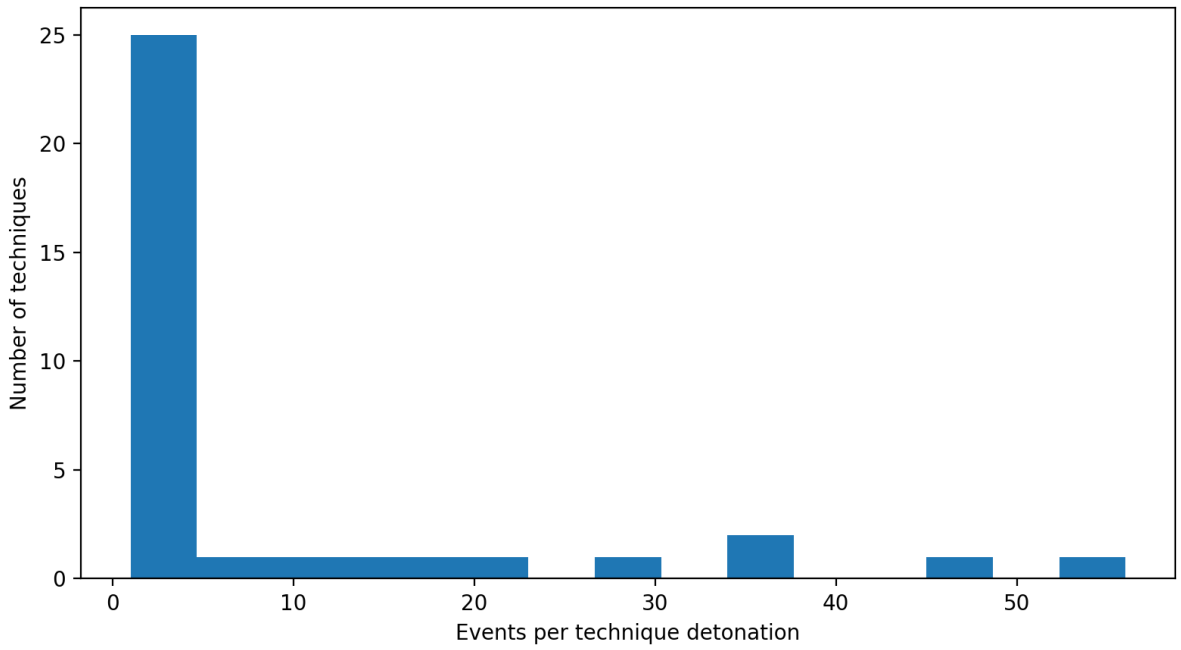


Fig. 3. Distribution of events per technique detonation trace.

3. Results and Discussion

We report three categories of empirical results: (i) event-level tactic classification, (ii) windowed and technique-level sequence recognition, and (iii) end-to-end attack-chain segmentation with explainable reporting. Across all experiments, the dataset and feature encoding described in the Method section are used consistently, and hyperparameters are fixed (random seed = 42) for reproducibility.

Dataset characteristics and class imbalance. Table II and Fig. 2 show that execution and credential-access account for most events (123 and 115 events, respectively). In contrast, exfiltration

DOI: 10.33474/infotron.v6i1.24923

Informatics, Electrical and Electronics Engineering (Infotron)
Volume 6, Number 1, (2026) pp. 28-43
ISSN 2798-0197

contributes only four events across four techniques, and initial-access and privilege-escalation are represented by one event each. This imbalance reflects the fact that some cloud techniques trigger many API calls (e.g., remote command execution through SSM) while others are implemented by one decisive call (e.g., a single bucket-policy modification). Imbalance is a realistic property of audit logs, but it affects supervised learning: sparse tactics have limited representation for the classifier and dominate macro-averaged uncertainty. In the current study we partially mitigate this with balanced class weights and by emphasizing macro-F1 and per-class F1; however, more robust treatment will require larger datasets and possibly resampling or class-balanced objectives.

Event-level tactic classification. Table IV reports 5-fold stratified cross-validation performance on 310 events. LinearSVC with TF-IDF achieved the best macro-F1 (0.866) and accuracy (0.961). Logistic regression performed similarly (macro-F1 0.832, accuracy 0.935), while naive Bayes lagged (macro-F1 0.675). The majority baseline reached only 0.397 accuracy and a macro-F1 of 0.069, confirming that the task requires semantic features beyond class frequency. Because the evaluation is conducted on a small, controlled dataset whose techniques often contain distinctive service-operation tokens, these numbers should be interpreted as baseline results on public detonation logs rather than as direct estimates of performance on noisier operational environments.

Confusion analysis. Fig. 4 shows the confusion matrix for the best event-level model (LinearSVC trained on all events). High-support classes are recognized with near-perfect precision and recall, especially execution and credential-access. Errors concentrate in low-support tactics. The single initial-access event was predicted as credential-access, and the single privilege-escalation event was predicted as persistence. Fig. 5 highlights this effect: initial-access and privilege-escalation have F1 = 0 because there is no opportunity to learn robust discriminators from one example. For defense-evasion and persistence, most errors correspond to events that are common across administrative workflows (e.g., IAM policy and role operations).

Per-tactic interpretation. Even when a tactic is well recognized, the explanation must be stable under benign variation. We examined the top-weight tokens for the dominant tactics. For execution, the highest-weight tokens correspond to SSM service operations and request schema keys. For credential-access, the model assigns weight to API calls related to key retrieval and credential material. Because these tokens are derived from CloudTrail fields, they align with analyst intuition: they describe which AWS service was used (ssm, iam, ec2) and which sensitive operation occurred.

Comparison to complex models. The strong event-level performance of LinearSVC indicates that the current dataset contains linearly separable signatures at the token level. However, because we did not benchmark CRF-, LSTM-, or transformer-based sequence models on this small corpus, the present results should be interpreted as establishing a transparent baseline rather than as demonstrating superiority over higher-capacity alternatives. The practical advantage of the chosen models is that their decisions and transition assumptions remain directly auditable [11], [12], [18].

Technique identification. We next evaluate whether short CloudTrail sequences can be attributed to specific Stratus techniques. We generated 25 augmented sequences per technique (875 total) and performed a stratified 80/20 split. Table V shows that both linear models achieved high accuracy: LinearSVC reached 0.989 accuracy and 0.988 macro-F1, while one-vs-rest logistic regression reached 0.983 accuracy and 0.982 macro-F1. These results are consistent with the dataset design: technique traces contain distinctive service-operation patterns, and TF-IDF n-grams capture these patterns directly. Because augmentation only deletes or duplicates existing events, it does not introduce spurious identifiers or unrealistic operations.

Technique identification error modes. The small error rate in Table V corresponds to confusion among techniques that share similar service-operation subsequences. For example, techniques that manipulate IAM policies and roles can share overlapping calls. In these cases, the explanation is still valuable: even when technique identity is ambiguous, the model surfaces the shared evidence tokens, enabling an analyst to inspect the raw events and disambiguate based on request parameters or resource targets.

DOI: 10.33474/infotron.v6i1.24923

Informatics, Electrical and Electronics Engineering (Infotron)
Volume 6, Number 1, (2026) pp. 28-43
ISSN 2798-0197

Windowed tactic classification. Table VI reports 5-fold cross-validation on 248 sliding windows of length four events. LinearSVC achieved 0.811 macro-F1, while logistic regression achieved 0.625 macro-F1. The window task is harder than the single-event task for two reasons. First, windows mix distinctive and generic events; generic events dilute discriminative signals. Second, the window label is inherited from the technique's tactic even if some constituent events are not strongly tied to that tactic. Despite this, the window classifier provides a useful intermediate representation for systems that batch events (for example, hourly audit summaries or sampling-based pipelines).

Attack-chain segmentation. The primary goal of this work is to recover a coherent stage timeline from an event stream. Table VII reports results on 60 synthetic multi-stage chains constructed by concatenating technique traces. The baseline strategy (LR independent) assigns each event the argmax tactic from the logistic regression classifier and derives boundaries whenever the predicted label changes. This achieved 0.840 accuracy, 0.873 macro-F1, and 0.800 boundary-F1. Applying HMM smoothing with Viterbi decoding (LR + HMM) substantially improved boundary detection to 0.892 boundary-F1, while accuracy and macro-F1 increased slightly to 0.848 and 0.877, respectively. These results suggest that sequence modeling is particularly helpful for suppressing spurious label flips in this controlled setting, although the gain should not be over-generalized to mixed benign/malicious production streams without further validation.

Why HMM smoothing helps. The HMM combines two sources of information: emissions (per-event probabilities) and transitions (stage continuity). A single ambiguous event with a weak emission can be overridden by strong continuity evidence from surrounding events, which prevents short-lived stage flips. At the same time, the transition matrix is learned from chains and therefore reflects the empirical ordering frequency in the dataset. Because transition probabilities are explicit, an analyst can audit them and adjust priors if a different stage ordering is desired.

Visualization of stage timelines. Fig. 6 visualizes an example chain's predicted tactics across event indices. The independent classifier exhibits local deviations that fragment segments. HMM smoothing removes many of these deviations by preferring plausible transitions supported by the transition model. In operational terms, this reduces alert fatigue: instead of generating multiple stage transitions due to single-event misclassifications, the system generates a stable timeline that better matches analyst expectations.

Ablation on interpretability features. Table VIII quantifies how different feature subsets contribute to event-level tactic recognition. Using only the API operation name (name) produced 0.516 macro-F1. Adding the service namespace (src+name) increased macro-F1 to 0.663. Adding principal type (src+name+user) achieved 0.665 macro-F1. The best setting (all_core: src, name, userType, readOnly, managementEvent) achieved 0.688 macro-F1. These results show that most discriminative power comes from service and operation names, while a small set of boolean flags improves robustness. This is important for compliance: these features are stable, explainable, and do not depend on sensitive identifiers.

Runtime and deployability. Table IX reports training and inference costs for LinearSVC and multinomial logistic regression on the full 310-event dataset. Both models train in under 0.03 seconds and classify an event in approximately 0.01 ms on the evaluation hardware. This efficiency enables deployment in stream processors and serverless pipelines, where latency and cost constraints are strict.

Explainability and report semantics. Fig. 7 provides a representative explanation template for the execution tactic, showing the highest-weight tokens from the logistic regression model. The top-ranked features include SSM-related signals and request schema keys associated with command execution. In an alert report, these tokens are rendered back into CloudTrail field names and paired with the concrete evidence events (timestamp, eventSource, eventName, errorCode, and schema keys). This yields a verifiable rationale: an analyst can cross-check the listed CloudTrail records and confirm that the predicted stage corresponds to remote command execution behavior.

Case study: generating an audit-friendly alert. We applied the full pipeline to the example synthetic chain used for Fig. 6, which contains six consecutive tactics in the ground truth: execution,

DOI: 10.33474/infotron.v6i1.24923

Informatics, Electrical and Electronics Engineering (Infotron)
Volume 6, Number 1, (2026) pp. 28-43
ISSN 2798-0197

privilege-escalation, credential-access, discovery, exfiltration, and impact. The execution segment begins with ec2:RunInstances and sts:AssumeRole events, followed by an iam:UpdateLoginProfile privilege-escalation action. The pipeline then identifies a long credential-access segment dominated by ec2:GetPasswordData calls, a discovery segment containing ec2:DescribeInstanceAttribute operations, a single-event exfiltration step (ec2:ModifySnapshotAttribute), and an impact segment comprised of Amazon Bedrock operations. Using LR + HMM decoding, the stage sequence is smoothed into contiguous segments and the boundary set matches the concatenation points more closely than independent classification (Table VII). The generated report lists representative evidence records for each segment (timestamp, eventSource, eventName, and key request schema tokens) and surfaces the highest-weight linear features that support the stage decision. This case study demonstrates that the report remains actionable even when techniques are short: a single privileged action or exfiltration API call is preserved as its own segment and is presented with explicit CloudTrail evidence and ATT&CK mapping.

Threats to validity. The dataset represents controlled detonations rather than production workloads, and it does not include benign background traffic interleaved with attacks. Synthetic chains concatenate technique traces without realistic time gaps or concurrent benign events, and they are derived from the same public corpus used to define the feature space and evaluation tasks. These design choices isolate the staging problem and make the experiments reproducible, but they simplify real-world noise, overlap, and domain shift. Nevertheless, the evaluation captures a common SOC workflow: after upstream filtering (rules, anomaly scores, or triage), an analyst reviews a condensed set of relevant events and needs a coherent attack narrative. The proposed approach targets exactly this step, but additional validation on mixed operational streams remains necessary.

Deployment considerations. In production, CloudTrail volume and coverage vary by account and configuration. The feature representation used here is intentionally robust to anonymization and cross-account centralization, but it assumes that key CloudTrail fields are present. For example, if only management events are collected, data-plane exfiltration signals may be missing. From explanations to detections. A practical byproduct of linear explanations is that they can be converted into detection engineering artifacts. For example, the highest-weight tokens for execution often include (src=ssm.amazonaws.com, name=SendCommand) and request schema keys that indicate command delivery. These tokens can be rendered into a Sigma rule condition or a SIEM query with explicit CloudTrail field predicates. Similarly, high-weight defense-evasion tokens surface operations such as StopLogging or UpdateTrail, which map directly to audit controls and to high-severity alert categories. This conversion step does not require re-training: it uses the already-learned coefficients as a ranked suggestion list for rule authors. In this sense, the model functions as a data-driven "rule recommender" that remains grounded in verifiable CloudTrail evidence and can be governed through standard detection review processes.

The pipeline can be integrated into a larger system by combining stage recognition with existing detectors: rules can generate high-precision candidate events and the model can then summarize and stage the surrounding context.

Future work will extend evaluation to mixed benign and adversarial streams, larger CloudTrail corpora, and stronger sequence baselines such as CRFs, LSTMs, and transformer encoders while preserving explanation fidelity. Future work should also quantify explanation utility through analyst-oriented evaluation, for example by measuring time-to-triage, stage-verification accuracy, inter-analyst agreement, and perceived usefulness. The present results therefore position the linear + HMM stack as a transparent baseline, not as the final state of the art for cloud attack-chain analytics.

Interpreting macro vs. weighted performance. Because the dataset is imbalanced, weighted-F1 is dominated by the most frequent tactics (execution and credential-access), while macro-F1 emphasizes robustness across all tactics. In Table IV, LinearSVC achieved weighted-F1 of 0.963 but macro-F1 of 0.866; the gap is driven by sparse tactics where single errors collapse the per-class F1 to zero. For deployment, this suggests a two-level evaluation practice: track weighted metrics to ensure the model

DOI: 10.33474/infotron.v6i1.24923

Informatics, Electrical and Electronics Engineering (Infotron)
Volume 6, Number 1, (2026) pp. 28-43
ISSN 2798-0197

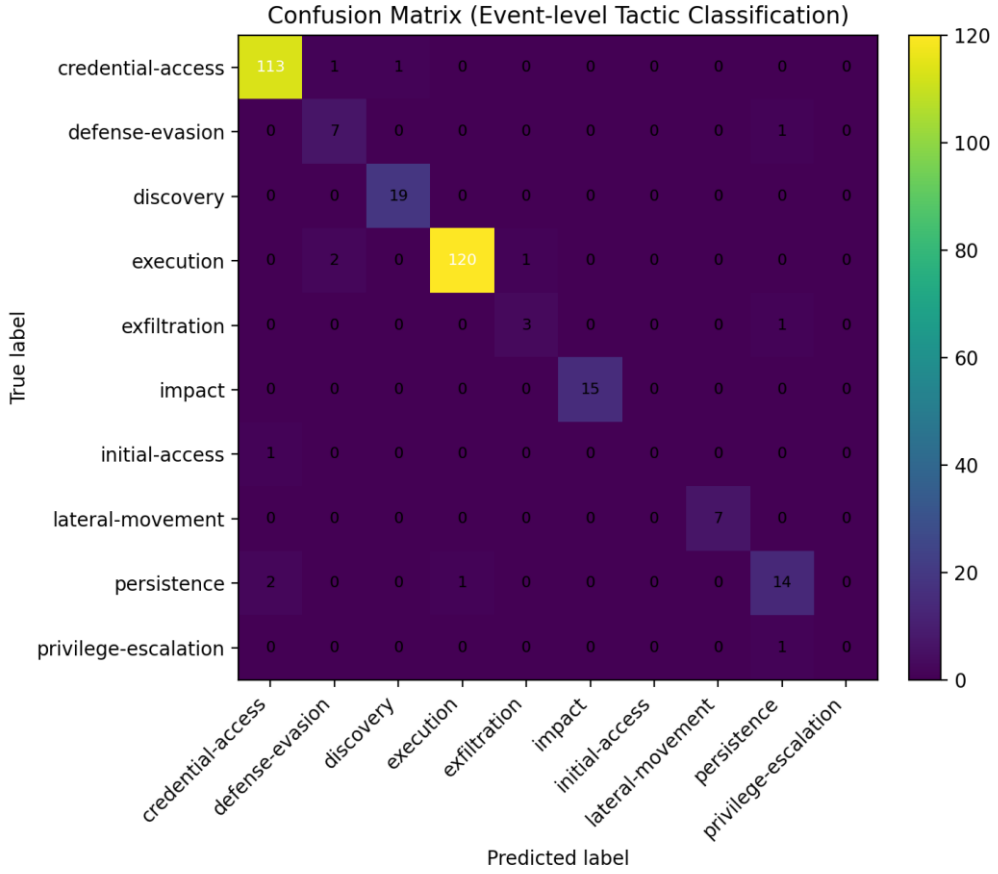
remains accurate for high-volume stages, and separately track per-tactic and macro metrics to ensure rare but critical stages do not silently degrade.

Boundary precision and recall. Table VII decomposes boundary quality into precision and recall. The independent LR baseline achieved 0.817 boundary precision and 0.798 recall, indicating that it both introduced spurious boundaries (precision < 1) and missed true transitions (recall < 1). After HMM smoothing, boundary precision increased to 0.900 and recall increased to 0.885. This improvement matches the intended role of temporal modeling: to suppress short-lived label jitter (raising precision) while preserving true stage changes supported by a sequence of events (raising recall).

From explanations to detections. The linear explanations produced by this pipeline are not only human-readable; they can be used to bootstrap deterministic detections. For instance, if the execution stage is consistently explained by tokens such as src=ssm.amazonaws.com and name=SendCommand, a detection engineer can implement a high-precision rule that flags SendCommand actions initiated by unexpected principals, and then rely on the model to stage surrounding context. Similarly, a defense-evasion explanation containing CloudTrail StopLogging or DeleteTrail can be translated into an always-on alert rule, while the model provides the narrative that connects logging disruption to subsequent credential or execution activity. In this way, explainable sequence learning complements (rather than replaces) traditional detection content.

Table 4. Event-level tactic classification (5-fold CV).

Model	Accuracy (mean)	Macro-F1 (mean)	Weighted-F1 (mean)
LinearSVC (TF-IDF)	0.96129	0.866321	0.95759
LR (TF-IDF)	0.935484	0.83217	0.940479
NB (Count)	0.909677	0.675488	0.890459
Majority	0.396774	0.069414	0.225465



DOI: 10.33474/infotron.v6i1.24923

Informatics, Electrical and Electronics Engineering (Infotron)
Volume 6, Number 1, (2026) pp. 28-43
ISSN 2798-0197

Fig. 4. Confusion matrix for event-level tactic classification (LinearSVC).

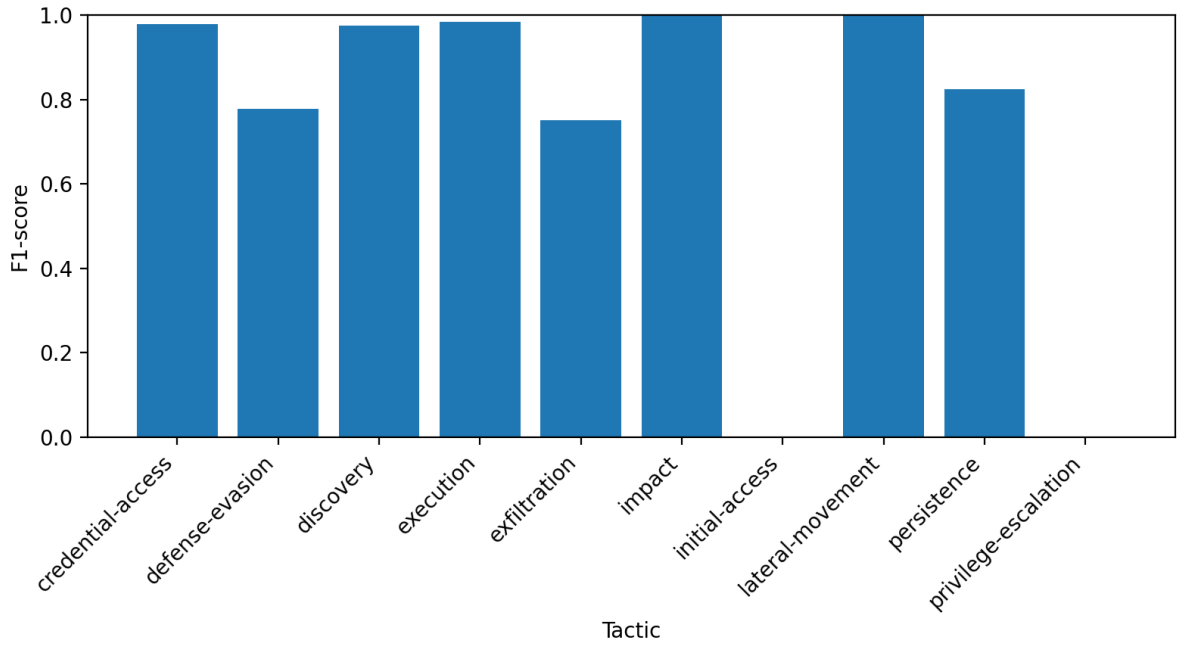


Fig. 5. Per-class F1 scores for event-level tactic classification (LinearSVC).

Table 5. Technique identification on augmented detonation traces (80/20 split).

Model	Accuracy	Macro-F1	Weighted-F1	Train samples	Test samples
LinearSVC (TF-IDF 1-2gram)	0.988571	0.988095	0.988095	700	175
LogReg OvR (TF-IDF 1-2gram)	0.982857	0.982323	0.982323	700	175

Table 6. Windowed tactic classification (window length = 4, 5-fold CV).

Model	Accuracy (mean)	Macro-F1 (mean)	Windows
LinearSVC (TF-IDF 1-2gram)	0.826694	0.810846	248
LogReg OvR (TF-IDF 1-2gram)	0.790449	0.624926	248

Table 7. Attack-chain segmentation on synthetic multi-stage chains (60 test chains).

Model	Accuracy	Macro-F1	Boundary-F1	Test chains
LR + HMM (Viterbi)	0.84775	0.876817	0.891667	60
LR (independent)	0.840014	0.872682	0.799544	60

DOI: 10.33474/infotron.v6i1.24923

Informatics, Electrical and Electronics Engineering (Infotron)
Volume 6, Number 1, (2026) pp. 28-43
ISSN 2798-0197

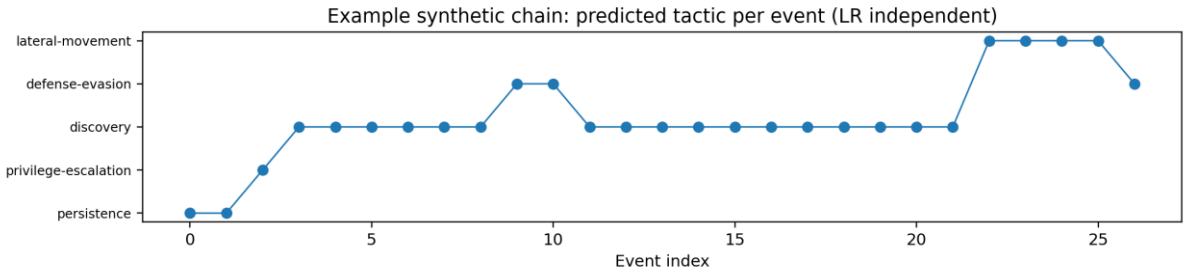


Fig. 6. Example predicted tactic sequence across a synthetic chain (independent LR predictions).

Table 8. Feature ablation for event-level tactic recognition (LR, 5-fold CV).

Feature set	Accuracy (mean)	Macro-F1 (mean)
all_core	0.758065	0.688439
src+name+user	0.770968	0.664913
src+name	0.76129	0.663253
name	0.690323	0.51589

Table 9. Runtime for training and per-event inference.

Model	Train time (s)	Per-event inference (ms)
LinearSVC (TF-IDF)	0.0121441	0.0106381
LogReg multinomial (TF-IDF)	0.0254889	0.0118494

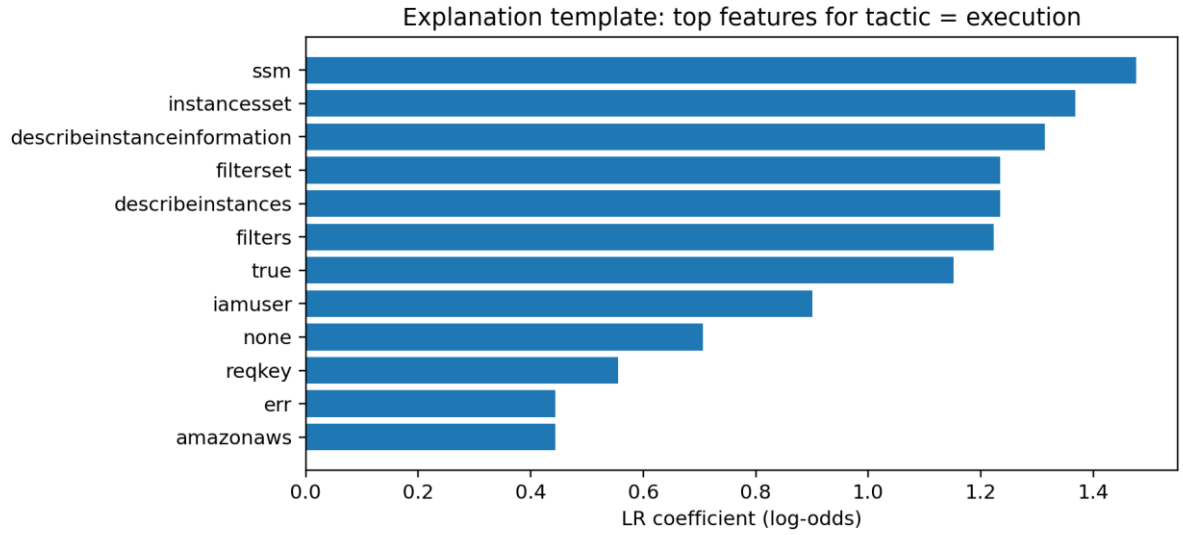


Fig. 7. Example explanation: top logistic-regression feature weights for the execution tactic.

4. Conclusion

This paper presented a reproducible baseline for interpretable attack-chain stage detection from AWS CloudTrail event sequences. Using the public Stratus Red Team detonation logs (v2.23.2), we showed that schema-aware, audit-friendly features combined with linear classifiers provide strong event-level and short-window tactic recognition on a small public dataset, and that adding an HMM with Viterbi decoding produces a more stable stage timeline in controlled synthetic-chain evaluation. Across 310 labeled CloudTrail events spanning 35 techniques and 10 tactics, LinearSVC achieved 0.961

DOI: 10.33474/infotron.v6i1.24923

Informatics, Electrical and Electronics Engineering (Infotron)
Volume 6, Number 1, (2026) pp. 28-43
ISSN 2798-0197

accuracy and 0.866 macro-F1 for event-level tactic classification. On 60 synthetic multi-stage chains, HMM smoothing improved boundary-F1 from 0.800 to 0.892, while accuracy and macro-F1 increased modestly from 0.840 to 0.848 and from 0.873 to 0.877, respectively. Beyond predictive performance, the pipeline generates coefficient-faithful alert reports that link each predicted stage to concrete evidence events and influential feature tokens. These findings suggest that transparent baselines can provide useful, auditable support for cloud attack-chain analysis, but the current evidence remains limited by dataset size, class imbalance, and the synthetic evaluation setting. Larger mixed-event evaluations and quantitative analyst studies are needed before stronger general claims are made.

In addition to its empirical performance, the approach is intentionally compatible with existing detection engineering. A team can treat model explanations as candidate rule primitives: high-weight service-operation tokens and request-key tokens can be promoted into Sigma-style detections or SIEM queries, while the HMM-smoothed stage timeline provides the contextual grouping required to reduce false positives. For organizations prioritizing governance, the same report can be stored as supporting evidence for security controls and post-incident reviews. In this sense, the system can support both detection workflows and documentation needs, helping bridge machine-learned pattern recognition and the requirements of human review.

5. References

[1] Amazon Web Services, "AWS CloudTrail User Guide," AWS Documentation, 2024.
[2] Amazon Web Services, "CloudTrail log file examples," AWS Documentation, 2024.
[3] Datadog, "Stratus Red Team (AWS detonation logs v2.23.2)," GitHub repository, 2025.
[4] Datadog, "Grimoire: CloudTrail Trickery for Red Teams," GitHub repository, 2024.
[5] Datadog, "LogLicker: CloudTrail Log Anonymizer," GitHub repository, 2024.
[6] MITRE, "ATT&CK Cloud Matrix," MITRE ATT&CK, 2025.
[7] National Institute of Standards and Technology, "Security and Privacy Controls for Information Systems and Organizations," NIST Special Publication 800-53 Rev. 5, 2020.
[8] National Institute of Standards and Technology, "Guide to Intrusion Detection and Prevention Systems (IDPS)," NIST Special Publication 800-94, 2007.
[9] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," Proc. IEEE, vol. 77, no. 2, pp. 257-286, 1989.
[10] C. Cortes and V. Vapnik, "Support-vector networks," Machine Learning, vol. 20, pp. 273-297, 1995.
[11] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in Proc. ICML, 2001.
[12] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in Proc. ACM CCS, 2017.
[13] S. He, J. Zhu, P. He, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in Proc. IEEE ICWS, 2017.
[14] W. Xu, L. Huang, A. Fox, D. Patterson, and M. Jordan, "Detecting large-scale system problems by mining console logs," in Proc. ACM SOSP, 2009.
[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in Proc. NAACL-HLT, 2019.
[16] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why should I trust you?": Explaining the predictions of any classifier," in Proc. ACM KDD, 2016.
[17] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in Proc. NeurIPS, 2017.
[18] Y. Liu, X. Wang, J. Wang, and H. Zhang, "LogBERT: Log anomaly detection via BERT," arXiv preprint arXiv:2103.04475, 2021.
[19] Open Cybersecurity Schema Framework, "OCSF Schema," OCSF Foundation, 2024.
[20] SigmaHQ, "Sigma: Generic Signature Format for SIEM Systems," GitHub repository, 2024.

DOI: 10.33474/infotron.v6i1.24923

Informatics, Electrical and Electronics Engineering (Infotron)
Volume 6, Number 1, (2026) pp. 28-43
ISSN 2798-0197

[21] R. Patil, S. Muneeswaran, V. Sachidananda, and M. Gurusamy, "E-Audit: Distinguishing and investigating suspicious events for APTs attack detection," *Journal of Systems Architecture*, vol. 144, Art. no. 102988, Nov. 2023, doi: 10.1016/j.sysarc.2023.102988.

[22] M. Keshk, N. Koroniotis, N. Pham, N. Moustafa, B. Turnbull, and A. Y. Zomaya, "An explainable deep learning-enabled intrusion detection framework in IoT networks," *Information Sciences*, vol. 639, Art. no. 119000, Aug. 2023, doi: 10.1016/j.ins.2023.119000.

[23] I. H. Sarker, H. Janicke, A. Mohsin, A. Gill, and L. Maglaras, "Explainable AI for cybersecurity automation, intelligence and trustworthiness in digital twin: Methods, taxonomy, challenges and prospects," *ICT Express*, vol. 10, no. 4, pp. 935–958, Aug. 2024, doi: 10.1016/j.icte.2024.05.007.

[24] B. Al-Sada, A. Sadighian, and G. Oligeri, "MITRE ATT&CK: State of the Art and Way Forward," *ACM Computing Surveys*, vol. 57, no. 1, Art. no. 12, Oct. 2024, doi: 10.1145/3687300.

[25] W. Lin, J. Ma, T. Li, H. Ye, J. Zhang, and Y. Xiao, "CrptAC: Find the Attack Chain with Multiple Encrypted System Logs," *Electronics*, vol. 13, no. 7, Art. no. 1378, Apr. 2024, doi: 10.3390/electronics13071378.

[26] M. Boffa, I. Drago, M. Mellia, L. Vassio, D. Giordano, R. Valentim, and Z. B. Houidi, "LogPrécis: Unleashing language models for automated malicious log analysis: Précis: A concise summary of essential points, statements, or facts," *Computers & Security*, vol. 141, Art. no. 103805, Jun. 2024, doi: 10.1016/j.cose.2024.103805.

[27] Y. Chen, M. Cui, D. Wang, Y. Cao, P. Yang, B. Jiang, Z. Lu, and B. Liu, "A survey of large language models for cyber threat detection," *Computers & Security*, vol. 145, Art. no. 104016, Oct. 2024, doi: 10.1016/j.cose.2024.104016.

[28] L. Ben-Shimol, D. Lavi, E. Klevansky, O. Brodt, D. Mimran, Y. Elovici, and A. Shabtai, "Detection of compromised functions in a serverless cloud environment," *Computers & Security*, vol. 150, Art. no. 104261, Mar. 2025, doi: 10.1016/j.cose.2024.104261.

[29] J. Ren and R. Geng, "Provenance-based APT campaigns detection via masked graph representation learning," *Computers & Security*, vol. 148, Art. no. 104159, Jan. 2025, doi: 10.1016/j.cose.2024.104159.

[30] Z. Weng, W. Zhang, T. Zhu, Z. Dou, H. Sun, Z. Ye, and Y. Tian, "RT-APT: A real-time APT anomaly detection method for large-scale provenance graph," *Journal of Network and Computer Applications*, vol. 233, Art. no. 104036, Jan. 2025, doi: 10.1016/j.jnca.2024.104036.

DOI: 10.33474/infotron.v6i1.24923