International Journal of Sustainability and Innovation in Engineering (IJSIE)
https://www.doi.org/10.56830/IJSIE2024

Vol. 2 2024

API Gateway Threat Prevention in Large-Scale Applications

Ramanan Hariharan
Lead Senior Manager, IAM Engineering, San Francisco, USA
Email: email@ramananhariharan.com

Received: 5 June 2024. Accepted: 10 August 2024. Published: 9 October 2024

Abstract

WAFs and API gateways are the lowest common denominator in every transaction and, therefore, the logical place to have the backend integrate threat prevention that is made scalable. This paper focuses on proactive, synchronous measures: driver block, driver challenge, rate-limit, sandbox, and step-up reauthentication, as opposed to detection procedures. It proposes an end-to-end design that combines on-path risk ratings with a deterministic policy engine and strictly enforced tail-latency budgets. This evaluation is capable through the use of a privacy-preserving, multi-region month-long corpus (~10B requests) in the form of gateway logs, auth events, WAF flags, and honeypot hits. Only O(1) online operations are supported: the presence of the header, the existence of specific trie paths, the length and entropy of tokens and parameters, rarity statistics, and sliding-window counters based on hashed client and tenant identifiers. The available serving options include in-process WebAssembly or a gRPC scorer with stringent deadlines; isotonic calibration and per-endpoint thresholds (including hysteresis) are available that map risk space to action. Idempotent GET caching, fail-open/closed defaults based on endpoint criticality, and signed audit logs can be used to provide reliability and governance. Offline experiments include traditionally separated splits PR-AUC and recall at specific false-block rates; ablations measure feature and model contributions. Online shadow/canary trials can be used to decrease malicious acceptance rate without increasing latency p95/p99 by more than ≤5-10ms. With 10^5-10^7 requests per second, between tenants and across regions, the strategy achieves SLOs and signed, hot-reloaded policy/model bundles. The artifacts consist of a public schema, feature definitions, a synthetic generator, policy DSL examples, rollback playbooks, config, and release scripts to provide reproducible deployment.

Keywords:

API gateway, Threat prevention, Real-time risk scoring, Policy enforcement, False Block Rate (FBR).

1. Introduction

In microserver architecture, gateways are positioned on the critical path of every request and mediate authentication, routing, observability, and security between untrusted clients and backend services. The gateway, in control-plane terms, takes configuration such as routes, schemas, rate limits, and policies, and grows signed bundles to data-plane instances. The per-request filters are deployed in the data plane and must meet narrow tail-latency constraints and add only a few milliseconds to total latency. Since all traffic enters the gateway, it is also in the perfect position to monitor cross-service behavior, compute light features, and apply risk-taking policies in an applicable manner. The modern threat environment takes advantage of this choke point: credential stuffing based on a login endpoint, injection scans that cover parameter space, scraping by residential proxies, token thumping, and exploitation of unauthenticated public APIs. At cloud scale, systems need to support 10^5-10^7 requests per second across many regions and tenants, often behind anycast routing, and maintain zero downtime. Such environments need deterministic failure policies, rapid hot-reload of configuration, horizontal elasticity, and precise separation of control- and data-plane interests. They also need privacy-preserving telemetry- hashed identifiers, path templates instead of raw URLs, and coarse ASN geo-location- to remain effective without violating data-minimization requirements.

This research deals with proactive prevention of threats and not post hoc Detection of the same. Prevention refers to synchronous measures (block, challenge, rate-limit, sandbox, or step-up reauthentication) that are taken before a request is relayed downstream. An asynchronous notification of alerting and logging, which can be thought of as delaying the response to the request, is called Detection. The goal is to achieve maximum malicious traffic recall at a limited FBR with an acceptable latency service level. Concretely, additional p95 latency can be as much as 5-10 ms on the critical path, and p99 latency should be bounded, and additional head-of-line blocking should be avoided. The scope presupposes TLS termination at the gateway and access to the header information, method, normalized path template, the structure of query parameters, the classification of responses, rough user agent hash, and autonomous system number information. Bodies of raw payloads and personal data are out; features must be made out of metadata and timing. Undertakings made by the prevention layer must integrate with routing, retries, and idempotency practices, and must achieve graceful degradation under dependency loss. Rate-limit primitives/challenge endpoints are assumed to be available within the same point of presence, and circuit-breakers provide local fail-open or fail-closed behaviour, depending on the endpoint risk.

Three research questions guide the work. It addresses configurations of on-path risk scoring and policy enforcement that attain reliability and latency at scale. Examples of evaluated patterns are in-process models to WebAssembly, sidecar gRPC scorers with strict deadlines and circuit breakers, and deterministic policy engines with

learned thresholds. This study examines which categories of features provide quantifiable lift with little leakage: sliding-window counters keyed by hashed client identifiers, endpoint rarity statistics, parameter entropy and distribution, token-verification results, ASN reputation, and request burstiness. It also looks at how to apply per-endpoint thresholds and score mapping to endpoint actions to trade off precision, latency, and operational expense. Contributions include a dataset recipe and governance approach, online feature-store design suitable for on-path use, a temporal validation protocol that avoids leakage, and an online evaluation playbook including shadow, canary, and rollback procedures. Additional contributions are calibration and decision-caching techniques that preserve idempotency, and reliability patterns to enable multi-region/multi-tenant operation.

The paper is organized into different sections. Section 2 summarizes the background and machine-learning solutions to API security, with a specific focus on latency- and cost-sensitive reporting. Section 3 details the dataset, preprocessing, visual analytics, the threat model, and the end-to-end architecture of on-path scoring. Section 4 describes real-time risk scoring and policy enforcement, the policy DSL, feature budgets, scorer interfaces, calibration, decision caching, and governance. Section 5 details offline and online experiments, ablations, robustness checks, and cost analysis. Section 6 covers trade-offs, reliability, privacy, and limits. Sections 7 and 8 provide a summary and conclusions of future work, and list the artifacts supporting reproducibility. Information fields about timelines and reviews can be skipped to speed up implementation.

2. Literature Review

2.1 Gateway/WAF/IDS Baselines

Large-scale Gateway-centric controls usually start with Web Application Firewalls (WAFs), intrusion detection/prevention systems (IDS/IPS), and Layer-7 filters that isolate requests before reaching backends. Detection engines scan over normalized request parts (i.e., decoded paths, canonical query strings, lowercase headers) and compare against carefully crafted SQL/NoSQL injection, path traversal, command execution, and deserialization probes. Pre-routing checks that guards guard include deny-listed parameters, header allow/deny predicates, and strict method-path contracts. A decision is enriched by reputation and threat-intelligence feeds in the form of autonomous system numbers (ASNs), geolocation clues, and known-bad netblocks. Rate limiting applies token-bucket or leak-bucket-based algorithms to limit bursts, as well as provide sustained throughput ceilings on identity, API keys, or client subnets. On TLS-terminating gateways, mutual-TLS policies, HSTS, and cipher assignment supplement remedies in the application layer by closing down the transport surface. With microservice estates, these controls are at the edge proxy, as well as intra-gateways that limit blast radius and prevent lateral movement in the event one service is probed.

As highlighted in the figure below, the heterogeneous graph attention architecture internally consumes the gateway events to produce nodes including client_ip_hash, account, device fingerprint, token_id, ASN, endpoint, and tenant, and edges including login, request_to, and token_failure. It incorporates the node-level attention on projected node types, meta-path guided aggregation, and semantic-level attention to model coordinated behaviors (***Mei; Pan, & Liu, 2022)***. Risk-aware embeddings are then concatenated together and fed into an MLP to produce a prediction. This allows addressing the threat of coordinated abuse through the modeling of degree bursts, community structures, and temporal patterns, and stream count-min or HyperLogLog sketches to provide scalable exposure of per-endpoint risk aggregation scores in real time.

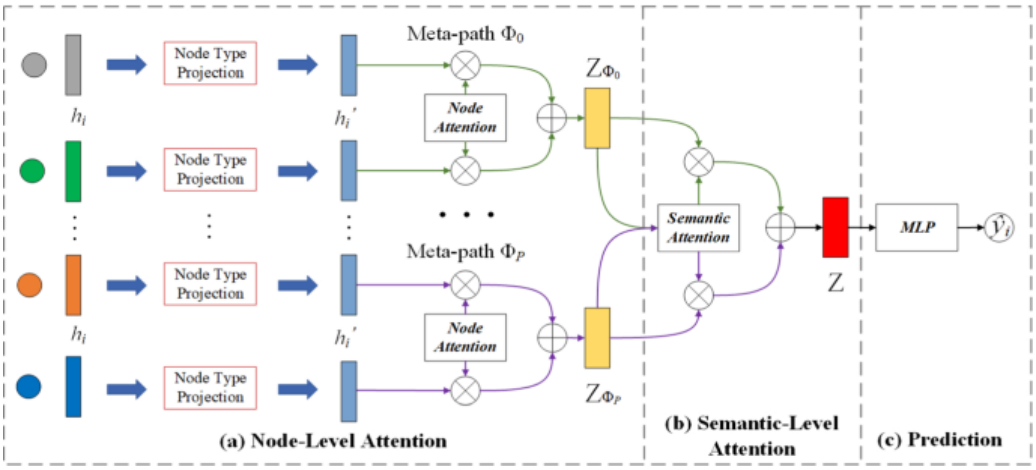

Figure 1: Heterogeneous graph attention pipeline for gateway event risk aggregation

A pragmatic internet-edge pipeline then enumerates countermeasures in cost-benefit priority: (1) request normalization to defeat evasion through encoding tricks, (2) inexpensive header and path filtering, (3) WAF rule evaluation that escalates to deeper parsing only as warranted, (4) reputation queries that have bounded latency and circuit breaker protecting the query with TTLs that expire on a negative-cache failure, and (5) token-bucket per-title checks that shed higher-rate routes. Placement counts: light-weight filters run on the gateway worker thread, but body inspections or reputation lookups run in asynchronous helper threads with hard time limits. It is critical to have configuration hygiene: the version of the rule is tracked, the rollout is canaried, and tenant and environment scope allowlists to prevent global weakening.

Advantages and shortcomings are obvious. Deterministic latency is present because, regardless of the evaluated rule or rate per state, they are constant-time and arithmetic on counts, therefore understandable to the decision justification by a specific rule or bucket state. Emergency mitigation is simple since operators can send a rule hotfix without modifying application code (***Repanovici; Nedelcu; Tarbă, & Busuioceanu, 2022).***. On the other hand, endpoint granularity and parameter diversity increase brittleness; header permutations and polymorphic payloads defeat signatures,

and reputation feeds decay rapidly and can cause regional false positives. The maintenance cost of control increases as the rules fall out of sync with the quickly evolving topologies of event-driven microservices, where both asynchronous flows and message contracts, as well as choreography, open new attack surfaces not captured by generic edge signatures. As a result, baseline controls alone cannot be used with modern event-driven systems. They must be supplemented by adaptive scoring that reasons not about payload strings but about behavior.

2.2 ML for API Security

Machine learning complements baselines by learning traffic behaviour as opposed to matching a fixed signature. Supervised formulations include binary and multilabel classifiers that score requests or short sessions based on features including method-path templates, the presence and absence of headers, parameter entropy and length histograms, bankrupt-token streaks, ASN rarity metrics, TLS/cipher metadata, and inter-arrival times. The use of O(1) computations and integer counters is highlighted in feature extractions to fit within a strict p95 latency budget on a gateway data path (***Chiariotti; Kucera; Zanella, & Claussen, 2019)***. Anomaly detection can also be used to supplement the supervised models in use where there is scanty label data: density estimation or reconstructor-error models can signal when behavior falls out of high-traffic patterns without prior characterization, intercepting probes that the rules do not. Hybrid systems use a combination of both: as a low-latency guardrail, reputation is used to take preventive and corrective actions, and a learned model can then provide risk scores that can then be consumed by a policy engine that maps those scores to actions (allow, rate-limit, challenge, block).

The quality of labels is also a limitation. Positives are analyst-confirmed incidents, honeypot hits, or backfills on incidents after the fact; negatives are sampled across both time and region with de-duplication of burst sequences. This adds noise to the process with delayed adjudication, inability to have 100\% ground truth, and feedback effects of the existing rules that have already acted as traffic filtering mechanisms. Classical imbalances - typically, positives are very rare (usually, below 1%) - require the use of cost-sensitive objectives, focal losses, or thresholds calibrated at specific false-block-rate (FBR) thresholds. The thresholds should be endpoint and per-tenant, as this takes into consideration different risks and impacts on business. Platt/isotonic Calibration biases scores toward match observed frequencies, allowing scores to reason at levels that are more discernible than opaque logits.

Diversity and redundancy are advantageous forms of supply-chain risk thinking: they cushion correlated failure. The equivalent in security scoring and telemetry is dual sourcing, using independent suppliers of key components. Interoperability among heterogeneous model families, a hot-standby scorer, and a mix of multiple intelligence feeds eliminates the single supplier blind spot and counters supply-based drifts. Practically, this means obvious forms of ensemble-level tactics such as voting or weighted averaging, parallel deployment of scorers on different

failure domains, and graceful degradation failure paths that drop to rules on poor models or feeds. This type of design reflects a dual-sourcing concept when the small cost is replaced with the elevated availability and the reduced systemic risk at the decisional level.

2.3 Real-Time/Streaming ML at Scale

When as many as tens of thousands to millions of requests per second occur, feature computation has to be incremental, cache-friendly, and predictable. Moving-window counters measure the number of requests per entity, the number of unique paths touched, the number of distinct user-identifying agents, bursts of error codes, and cross-tenant streaks of credential failures. These counters are keyed by several pairs, such as (client IP/MD5, client user agent /MD5, tenant ID). Correlated activity windows Grouping to identify low-and-slow credential stuffing or catalog scraping that may fall below individual-second thresholds. For stores with read-after-write, stores are optimized with ring buffers and approximate distinct counters to avoid per-request heap amortization with O(1) amortized writable updates. There is Hot-key protection by means of LFU/LRU-caches, tenant/entity-scoped circuit breakers, and shzetnTAyn strata.

A streaming model of serving demands with complex budgets and deterministic fallbacks. Executable Scorers have been compiled to native code (or WebAssembly), and run in-process on NUMA-pinned threads with pre-allocated arenas to stabilize p95 and p99 latencies. Scorers that operate in another process communicate with each other via gRPC with timeouts under 2-4 ms and circuit breakers that open when there are sudden changes in timeout counts—decision caching (***Larsson; Tärneberg; Klein; Kihl, & Elmroth, 2021)***. By memorizing actions of idempotent GETs on request fingerprints, it is possible to reduce recomputation in tens of seconds. Observability provides per-hop histograms of latencies, score-distribution drift monitors, and sometimes burn-rate alerts based on an error budget. Canary workflows shadow-score a subset of traffic over several days to confirm stability before switching all traffic over; this rollback option is automatic via feature flags in the policy engine.

The streaming telemetry literature offers workable prototypes. The scenario of fleet telematics shows how an edge-generated event at a large scale with high volume can be downsized, corrected, and transmitted within bandwidth and latency limits, yet maintain safety-relevant signals. Similar techniques, including on-edge aggregation, event compression, prioritized transmission queues, and heartbeat messages, directly apply to gateway scoring: aggregation can be done before leaving a gateway, low-value messages can be delayed during congestion, and health beacons can continuously ensure that a scorer is healthy. Another insight that telematics brings is the importance of a crude geospatial or network context (e.g., ASN changes and route rarity) to identify anomalous trajectories; in gateway terms, sudden ASN or user-agent changes within a session are powerful indicators of risk. These operational patterns — edge summarization, priority scheduling, and constant health telemetry — are easily

portable to API security pipelines, which need millisecond-based decisions at scale (***Nyati, 2018)***.

2.4 Gaps in the Existing Literature

Three omissions predominate in the scholarship and field practice. The end-to-end integration into the network of the gateway path is seldom an issue of prior engineering (***Nardini; Sabella; Stea; Thakkar, & Virdis, 2020)***. Lots of research presents offline metrics but does not specify orchestration details, such as where the feature computation should be put (in-filter or sidecar versus remote service), how far a per-hop budget can stretch across parsing, feature extraction, scoring, and policy calculation, and supported failure semantics (fail-closed versus fail-open per endpoint criticality). The literature on incident response is focused on prepared playbooks, command structures, and continuity planning (***Shaked; Cherdantseva; Burnap, & Maynard, 2023)***. However, the relationship between those efforts and millisecond in-path controls remains poorly explained in API-security literature. In practice, there are limited assessments as to how a blocklist or model rollback can be implemented in an atomic manner across the network, and with the auditability and evidence compliant with the highest levels.

Reporting on latency and cost does not evenly follow the specification. Offline curves like ROC-AUC or PR-AUC do not reflect the tails or CPU-cache performance that are key to whether a gateway can maintain service level targets in load. Examples of practical reporting would incorporate p50/p95/p99 added latency-per-hop, cache hit ratios of feature fetches, scorer timeouts and retries, and cost per-m entire scored requests with a breakdown into CPU time, memory, and egress-to any out-of-process calls. The evaluations need to report on the enforcement effect-recall at set FBR under both canary and A/B protocols, as well as measure MAR (malicious acceptance rate) reduction over isolated classifier accuracy. As shown in Figure 2, ROC-AUC describes offline discrimination of a classifier, but does not include gateway realities: p50/p95/p99 added latency per hop, CPU-cache effects, and feature-store cache hit ratios, scorer timeouts/retries, and cost-per-million scored requests in CPU time, memory, and egress to out-of-process calls. Such curves should be coupled to enforcement outcomes such as the level of protected malicious acceptance rate (MAR) reduction at fixed FBR during canary and A/B experiments.

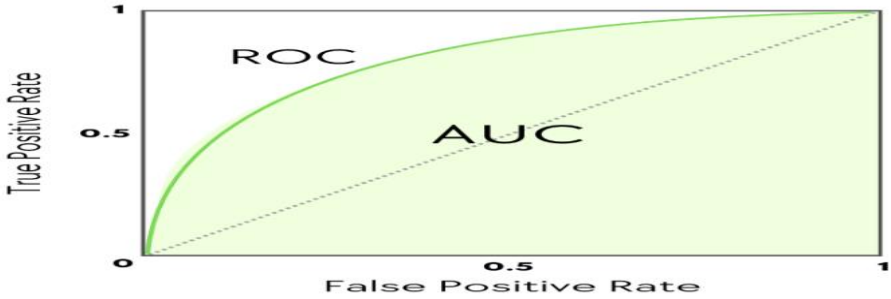

Figure 2: An example of ROC–AUC offline metric: insufficient for gateway latency–cost reporting

Privacy-constrained reproducibility is also a poorly developed area. Gateway datasets by necessity include sensitive identifiers and secret business context, which means that releasing them to the public is not permitted. Generalization, adversarial robustness, or drift tolerance cannot be confirmed without sharing the artifacts. The realistic following step is to publish workload-canonical but privacy-redacted schemas, synthetic data generators that preserve burstiness and endpoint coverage, feature-definition notebooks, and policy-DSL examples. Synthetic traffic with more realistic arrival processes could be replayed deterministically on the distributions of scores, the latency profiles, and action differences, which would allow a comparative study without jeopardizing the duty of data protection. Until then, much of the literature will be in the process of optimizing static measures that are disconnected from the operational realities of within-gateway decisioning.

3. Methods and Techniques

3.1 Description of Data Set

The API gateway is modeled as the common ingress of a multiregion, multi-tenant microservices domain, and an event corpus representing steady-state and promotion peak traffic is reconstructed over the course of a month. Motifs of five telemetry sources converge at the edge and coalesce into one per-request log: gateway access logs recorded at the end of the process, the authentication events sent by the identity provider and token-introspection sidecar, WAF event evaluation response, honeypot hits observed at decoy routes, and analyst-confirmed incidents uploaded to the case-management system. Records adhere to a compact schema that balances utility and privacy: {ts, client_ip_hash, asn, method, path_template, status, bytes, ua_hash, token_presence, tls_cipher, latency}. The ingest does not include free-text request bodies and request headers. The hash used to verify user-agent and IP values cryptographically is executed with per-region salts secured in a hardware-backed secrets manager that is rotated without causing downtime. ASN is looked up just once at the edges, and is cached as coarse-grained repute signals rather than as more precise locations (***Fainchtein, 2023)***. The columnar Parquet with Snappy compression is stored in an object-store hierarchy partitioned by dt, region, and service. Tiny files are compacted to 256-512 MB to speed up total scan throughput; a catalog stores partition

statistics and per-column min-max range to support predicate and column elimination. Daily deltas are echoed out to a low-latency analytics cluster to support near-real-time visualization without straining the archival lake. The corpus is intended to have approximately ten billion requests in thirty days and is also guarded by a retention window and access controls that are suitable for production telemetry.

Labeling is at the request and session levels. A request is tagged positive when targeting a honeypot route, when associated with an analyst-rated incident, or when it matches a short-horizon attack fingerprint identified by clustering combinations of status codes and parameter-entropy bursts. The sessions are estimated by grouping (client_ip_hash, ua_hash) with a window of twenty minutes of inactivity, and a positive request will elevate the session to positive. Negatives are strewn through time using a stratified (temporal) sampling rule to maintain signals at diurnal and weekly timescales, and to be unrepresentative of rest periods. To avoid feedback loops, characteristics that directly reflect previous enforcement effects are also not used during training, and any request that reflects previous blocking /challenging in the same session is discarded in the training set. Governance is implemented on an end-to-end basis: sensitive columns are encapsulated with role-based security rules, custom reads have purpose codes, and access is audited.

3.2 Data Preprocessing

Preprocessing begins with contract enforcement. Batches are rejected if required fields are absent or if values violate domain constraints (for example, method ∈ {GET, POST, PUT, PATCH, DELETE} and status ∈ [100, 599]). Numeric heavy-tailed variables--latency and bytes--are also winsorized at the 99.9th percentile, and transformed with log1p to stabilize variance and condition linear components. Categorical encodings are limited to reduce inference time and memory: method, tls_cipher, and coarse user-agent families are one-hot encoded with a special bucket; and path_template is target-encoded using K-fold cross-validation within the training time window to reduce inference latency and leakage. SN and token_presence are considered to be the low-cardinal features with explicit missing categories. Real-world contents are not stored literally; token count, length histograms, and entropy of parameter names are instead calculated to follow probe-like behavior without recreating sensitive data.

Table 1: Data preprocessing pipeline—key steps, parameters, and objectives

Step	Technique/Rule	Key Parameters	Purpose
Contract enforcement	Reject invalid batches	Required fields present; method ∈ {GET, POST, PUT, PATCH, DELETE}; status ∈ [100, 599]	Guarantee schema and domain integrity

Step	Technique/Rule	Key Parameters	Purpose
Numeric transforms	Winsorize + log1p	Winsorize latency/bytes at 99.9th percentile; apply log1p	Stabilize variance; condition linear components
Categorical encodings	One-hot + target encoding	One-hot: method, tls_cipher, coarse UA with “other”; path_template target-encoded via K-fold CV within time window	Reduce inference time; minimize leakage
Low-cardinality & missingness	Explicit categories/flags	SN, token_presence kept low-cardinality; tls_cipher="unknown"; binary flag for missing latency	Preserve signal without biased imputation
Privacy-preserving content	Summary statistics only	Token counts, length histograms, entropy of param names	Capture probe behavior without storing sensitive data
Sessionization	Group and timeout	Group by (client_ip_hash, ua_hash); inactivity timeout 5–30 minutes (endpoint-sensitive)	Define per-client windows for behavioral features
Rolling counters	Ring buffers, O(1) updates	Features: req_count_Δt, token_failures_Δt, unique_paths_Δt, mean_interarrival_Δt; windows: 30s, 5m, 1h	Low-latency online feature computation
Outlier suppression	Poisson thinning	Apply to exact-repeat micro-bursts	Prevent overweighting retries in training
Deduplication	Deterministic fingerprint	128-bit hash over {method, path_template, header_set, param_keys, body_size_bin}; collapse identical fingerprints within 5s per (client_ip_hash,	Remove replay artifacts; keep aggregated count

Step	Technique/Rule	Key Parameters	Purpose
		ua_hash)	
Packaging for serving	Ship preprocessing artifact	Encoders, winsor thresholds, ring-buffer params bundled with model	Ensure train/serve transformations are bit-equal
Promotion & rollout	Canary + blue-green	Compare distributional stats to baseline; check latency deltas against budgets before promotion	Safe, measurable deployment of preprocessing changes

The per-client windows for the Sessionsization construct are created based on the grouping of (client_ip_hash, ua_hash) and have an inactivity timeout of five to thirty minutes, with endpoint sensitivity used to tune this file. Within each session, rolling counters are computed using fixed-size ring buffers to guarantee O(1) updates: req_count_Δt, token_failures_Δt, unique_paths_Δt, and mean_interarrival_Δt for windows of thirty seconds, five minutes, and one hour. Outlier suppression suppresses micro-bursts of the exact requests through Poisson thinning in order to avoid overweighting retries during training. Deduplication addresses replay artifacts introduced by client or gateway retries: a deterministic request_fingerprint is built as a 128-bit hash over {method, path_template, header_set, param_keys, body_size_bin}; identical fingerprints from the same (client_ip_hash, ua_hash) within five seconds collapse to one row with an aggregated count. Missingness is treated clearly: tls_cipher is coded as “unknown” instead of the majority class; latency missingness is coded as binary with no form of central tendency bias occurring (***Paracha, 2023)***. The entire preprocessing graph embedded with encoders, winsorization thresholds, and ring-buffer parameters is an artifact deployed together with the model; thus, training and serving transformations remain bit-equal. Promotions are made on Canary traffic by measuring candidate artifacts in distributions against the baseline, testing distributional statistics and latency increments against budget thresholds, and staged by blue-green deployment to keep the prediction supply chain reiterating and otherwise efficient (***Kumar, 2019)***.

3.3 Data Exploration using Visual Analytics

Exploratory analysis is paired with near-real-time triage in batch retrospectives. Time-series tiles visualize HTTP status (2xx, 3xx, 4xx, 5xx), volume of requests, gateway processing times, and model prediction times every minute. Seasonal-trend decomposition isolates residuals; residuals more than three standard deviations define an anomaly that announces operator review. In authentication endpoints, an increased 401 / 403 ratio can be an indicator of credential stuffing activities, and not upstream

failure. Other tiles depict parameter-name entropy and header-set cardinality; spikes with high specificity in time can indicate injection probing of endpoint templates.

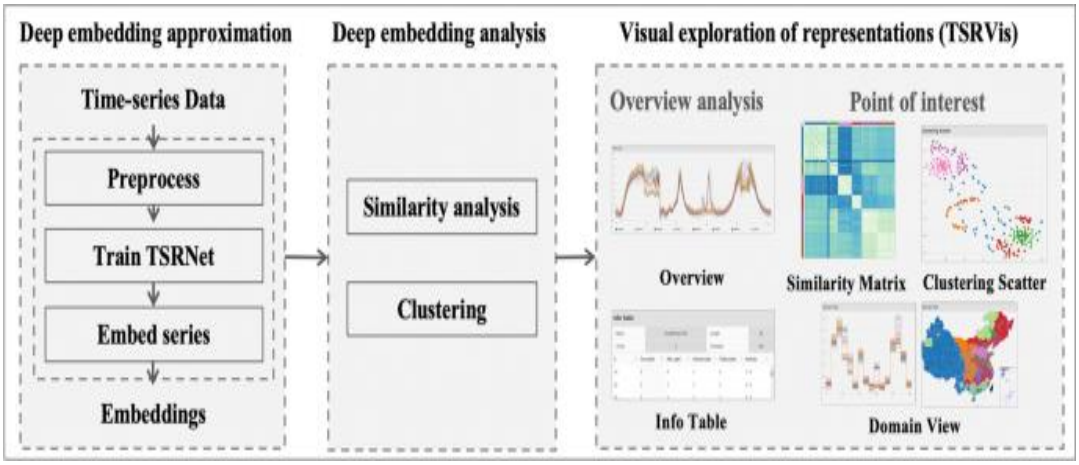

Figure 3: Time-series visual analytics for status, anomalies, and credential-stuffing signals

The workflow visual analytics enables triage and batch retrospectives in near-real time, as shown in the figure above. Time-series tiles present aggregated data of HTTP status cohorts (2xx, 3xx, 4xx, 5xx), request volume, gateway processing, and model prediction latency at the one-minute resolution. Seasonal-trend decomposition isolates residuals and deviations that exceed three standard deviations, which are reviewed as anomalies by operators. A trend in 401/403 ratios towards the high side is emphasized when characterizing authentication endpoint protection pressures (***Starc, 2023)***. Other tiles monitor parameter-name entropy and the cardinality of header sets; thin spikes in the former channel signal probing of templates. The similarity and clustering views enable one to accelerate root-cause analysis and facilitate actionable targeted mitigations by grouping the periods with shared signatures.

End data profiles. End profiles show the riskiest combinations to help adjust thresholds. Such as GET requests on POST /v1/token with concentrated 401/403 errors within a petite time frame would justify more aggressive policies than GET on cached catalog data that calls for a conservative policy. Raw ASN and region choropleths allow discerning volumetric changes devoid of granular geolocation threat; the unexpected spike of otherwise anonymously separate autonomous systems is often associated with hitherto unevocated bot campaigns or formerly exploited proxy networks. Each panel lets users apply interactive filters by tenant, endpoint family, or client fingerprint rareness decile to isolate behavior specific to a tenant or path cluster (***Balamurugan, 2023)***. Summary tables detail the per-endpoint FPR at candidate thresholds, and give the cumulative distribution of risk deciles, allowing rapid selection of operating points when both latency constraints and costs must be considered.

With false-positive and true-positive dashboards, the loop of learning is closed. The density graphs represent a risk_score distribution of TP, FP, and unlabeled traffic over endpoints. Overlap areas indicate a necessity in feature enrichment (such as inclusion of a short-window failure-rate counter) or endpoint-specific calibration. Drill-downs show an exemplar request with de-identified headers, the path template, grouped parameter counts, and a brief reason string constructed with the minimal list of features used in online processing. Cohort charts measure the novelty scores of new endpoints and new ASNs in order to decay aggregate novelty scores over a period of traffic maturation; cohorts suspected of anomalous behavior may be pinned in advance of enforcement ramping.

3.4 Threat Model & Attack Taxonomy

The threat model presupposes that adversaries have access to residential proxies, headless browsers, and low-cost orchestration to rotate IPs, mix headers, and execute some JavaScript to survive basic detection. Five classes of attacks are in scope. Credential stuffing manifests as bursts of attempted logins with recycled credentials re-used in previous breaches, identifiable by correlated token failure counters, low inter-arrival variance, and a peaking 401/403 ratio with no progressive backoff. Brute force Probes the issuance and validation paths; cluster identifiers with high entropy of names, repeated key, patterns, and 401 series along/v1/token paths. Injection probes appear as spikes and sustained activation of 400/404 with a high path entropy and infrequent header pairings. Scraping and botnets are broad rather than deep, with a wide coverage of path templates separated by homogeneous inter-arrival times and downstream error rates **(*Fajana, 2023)***. Temporary or permanent overuse of free tiers or trial endpoints will result in medium-rate traffic with repeatedly missing authentication markers but valid parameter shapes.

The defenses adapt to adversaries, who smear traffic across regions to avoid local rate controls, add minor delays to appear human, and replay subsets of valid headers used to bypass limited signatures. Defensive assumptions peg design decisions down. TLS is ended at the gateway and always reveals cipher suites and ALPN values; authentication middleware is homogeneous to the extent that a token validation semantics can be compared across services; rate-limit primitives can apply per-tenant and per-endpoint policies. The performance metric of attaining a low malicious acceptance rate (MAR) is throttled against a per-stakeholder-approved false block rate (FBR) cap. MAR and FBR are calculated on sampled production traffic using a combination of online decisions, analyst outcomes, and honeypot hits. They are set per endpoint families: a stricter budget for login and token endpoints, with a less stringent budget for catalog reads.

3.5 System Architecture & Feature Store

Incremental application of risk scoring is performed synchronously along the request path with an absolute limit of five milliseconds at the p95. Traffic passes

through the layer-7 proxy to terminate TLS and route to the guardrails, or allowlists, certificate requirements, and basic schema checks, which then connect to the risk scorer. Online features of the request and slide-window counters in an in-memory store are calculated, a compact model is evaluated, and a risk score is returned. The policy engine uses the score to determine the action, whether to allow, rate-limit, challenge, or block, according to each endpoint-specific threshold with hysteresis to prevent oscillation. To avoid tail latency, the scorer reveals a deadline; timeouts fail open on low-risk endpoints and revert to challenge on high-risk endpoints. A decision cache (request_fingerprint action) is then used and its contents cleared when policy or model versions change.

The store feature is designed to be predictable and updateable in real-time. Examples of keys include (client_ip_hash, ua_hash, tenant_id). BGs keep three-minute, five-minute, and one-hour rings of request counts and path templates that are unique (that is, where each input path belongs to at least four distinct path templates), and one-hour rings of token failure count and inter-arrival metrics. Updates increment the existing cell and move indices to the next level on epoch second. The rarity features exploit negative logarithmic frequency on a trailing horizon to emphasize unusual endpoints or shapes of parameters. The memory is limited by its per-key quotas and LFU eviction. All counters are regionally sharded so that updates are kept local; cross-region replication is eventual and used only to replicate analyst views, not synchronous enforcement. Warm-start preloads the last checkpoint when a process is started, so counters do not restart at zero following resource updates (***Song, et al., 2023)***. This is outstanding observability. The gateway exports p50/p95/p99 added latency of feature computation and scoring independent of the latency of the upstream services; score histograms by endpoint and tenant identify changes in calibration; and the stability of the population and Kolmogorov-Smirnov statistics compares the distribution of populations in training and live environments to identify data and feature drifts. Each decision has provenance--model version, feature-pack version, policy bundle, and input hash--so an operator can reproduce what happened. Canary rollouts are used to compare action distributions and added latency between incumbent and candidate bundles on a sample shard and gate promotion to error-budget burn rates.

4. Real-Time Risk Scoring & Policy Enforcement in API Gateways

4.1 Policy Engine Architecture & DSL

A direct implementation can be in the form of the policy engine directly on the API gateway hot path as an Envoy or Nginx filter compiled into WASM or Lua, or a sidecar that is colocated and joined synchronously via Unix domain socket. To maintain the user-facing SLOs, the policy call path can only add at most a few milliseconds at p95 to the latency-sensitive endpoints (***Qiu, H., Banerjee, S. S., Jha, S., Kalbarczyk, Z. T., & Iyer, R. K. 2020)***. The engine interprets a declarative policy written in a succinct DSL similar to OPA/Rego or CEL. Policies evaluate request

attributes (method, path_template, tenant_id), calculated risk_score, and the feature flags that open incremental policies.

To ensure predictability, the policy is represented in a deterministic directed acyclic graph with short-circuit evaluation; allowlists, mTLS-required, and hard denylists are the only rules giving a guardrail in that they always fire when matched. The policy bundles are signed, stored in an OCI-style registry where they are hot-reloaded with canary rollouts, and capability checks are triggered in case of an operator that is not known to or understood by the bundle. Each decision has a compliance hook that adds policy_version, rule_id, action, and an evidence_hash. Decision logs are chained (such as, hash-linked per window) to enable tamper-evident auditability appropriate to regulated situations and service-to-service attestations **(*Sardana, 2022)***.

4.2 Risk Scoring Pipeline & Feature Budget

The on-path scorer has a stringent feature budget, fifty or fewer, and a single allocation per feature, and would prefer to stick with O(1) operations to minimize jitter. Efficient primitives are header presence checks, trie-based path matching, token-length and character-class bounds, parameter-entropy estimates with thread-local histograms. Behavioral signals are maintained in an online store using LFU caches and ring buffers keyed by (client_ip_hash, ua_hash, tenant_id) to compute sliding-window counters such as requests/Δt, unique_paths/Δt, and token_failures/Δt. The gRPC scoring interfaces are typical: an in-process model (e.g., WASM-compiled linear or small tree ensemble), or a gRPC scoring interface with a hard deadline of two to four milliseconds and circuit breakers that open past 5% timeouts.

A feature schedule binds the risk-scoring pipeline to uphold latency SLOs as shown in the figure below. Planning and evaluation procedures assign budget slack over primitives: header presence checks, trie path matches, token-length bounds, and parameter-entropy histograms, and give it preference to primitives with O(1) cost. Behavioral counters are held through LFU caches and ring buffers indexed by client_ip_hash, ua_hash, and tenant_id. Models are in-process (WASM) or gRPC with 2-4 ms deadlines. Circuit breakers trigger above 5% timeouts to maintain performance-based budgeting, which helps enforce budgeting in fluctuating traffic and under distributions.

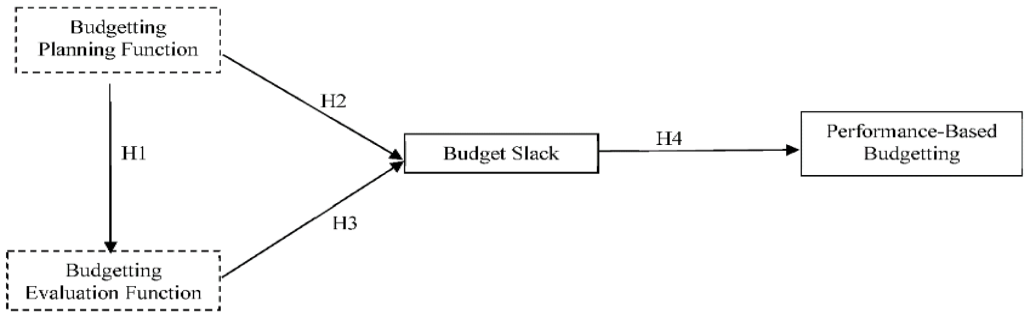

Figure 4: Feature budget planning for low-latency, performance-based risk scoring

Scores are calibrated on-gateway by a compact isotonic lookup to convert raw_score→risk_score; perspective constraints prevent safety features (invalid_signature, known_bad_asn) from increasing risk. On idempotent GETs, a request_fingerprint-based decision cache uses a 30-120 second TTL with an ETag of {policy, model} to auto-invalidate when dirty. The general architecture resembles that of memory-augmented inference pipelines: recent context is cached and consulted expressly to stabilize streaming-load decisions (***Raju, 2017)***

4.3 Enforcement Actions & Decisioning

The enforcement plane has to scale upwards and downwards between benign and hostile floods whilst preserving legitimate traffic. Available actions include ALLOW, RATE_LIMIT, CHALLENGE (CAPTCHA or proof-of-work), REAUTH (step-up OAuth or client mTLS), BLOCK, SANDBOX (degraded content), and QUARANTINE routing to a containment cluster. Risk to action mapping applies a hysteresis policy curve: as soon as risk exceeds an upper limit, the action will persist until the risk falls below a lower limit. It is tenant-, and endpoint-proficient, with stricter thresholds on a login endpoint than on a catalog endpoint because fraud costs are different (e.g., acquiring a footwear outfit would cost more than developing a pair of shoes). Nondetailed rate limiting may be parameterized dynamically based on risk_score: burst and sustained token-bucket rates are decreased as risk increases, and per-path_template limits. The challenge orchestration issues a 302-challenge service, and it passes an opaque nonce bound to (client, endpoint, time) to prevent relay; the replay window is a minute, and the challenge is single-use. Evidence packaging only contains the bare bones of the facts, necessary to analyst tooling (asn, failed_token_count, param_entropy) and redacts raw PII explicitly. The idea of thresholds and actions being personalized per tenant reflects the principles of customizing the guidance provided by AI to the context of a user and consequently improving the user outcomes within the limits of operating within the lines.

4.4 Latency, Reliability & Failure Modes

Latency and failure are first-eny design inputs. Per-hop budgets are precise: feature evaluation <=1.5 ms, scoring <=2.0 ms, policy evaluation <=0.5 ms, resulting

in the addition of a p95 of <=5 ms and a p99 of <=10 ms. Continuous profiling applies production-like traffic, synthetic tail amplification, and cache-cold conditions to reveal allocator contention, lock convoys, and head-of-line blocking. The Fail policy is endpoint-sensitive: several endpoints fail open to ALLOW on unavailability of the scorer; other endpoints fail closed to CHALLENGE or drop. Examples of resilience techniques include bounded queues with backpressure, 429 emission on overload, lock-free counters, and NUMA pinning of the scorer threads under extreme throughput. On a scorer timeout rate or error-budget burn, circuit breakers will trip into rules-only mode. Synthetic heartbeat requests track end-to-end time as well as decision rates; burn-rate alerts (e.g., 2x slo and 6x slo) spur on-call response. The actions of deferring synchronous scoring due to idempotency constraints cannot rely on retries to prevent reordering; unfavorable rulings are cached pessimistically to mitigate the blast radius caused by short-term misrepresentations. Decisions are assigned a decision ID to identify them in logs and tracked in metrics as well as ticketing systems, so they can be retraced to provide a lossless recon of the incident and be rolled back at a certain point.

4.5 Multi-Region, Multi-Tenant Governance

Global business means disciplined control and isolation. A highly unified control plane (Raft/ etcd) distributes region-scoped policy and model bundles signed with provenance, and rollouts are advanced behind health gates coupled with a per-region pause and resume. Sliding-window counters need to agree with points of presence; CRDT registers (commutative) or regular reconciliation of deltas per-PoP allows this to happen in a manner that avoids double-counting and skew. Isolation of tenants is achieved via namespacing of counters, features, and policies; pre-ML QPS budgets guard against tenants starving standard capacity. Data sovereignty is addressed through an input limit on features by geography, hashing or scrubbing of sensitive feature headers to the edge, and regionally-specific training when the availability of features varies. Migrations shadow-evaluate sampled traffic on the old or new policy traits and export applicable diffs when action distributions vary more than a configured tolerance. Auditing and analytics provide summary action histograms, risk deciles, false-block rate (FBR), and malicious acceptance rate (MAR) on a per endpoint and tenant level; signed, tamper-evident decision logs feed a central SIEM. Such behavior codifies into repeatable runbooks: runbooks to exercise shadow validation, canary with strict rollback constraints, score drift, and action delta monitoring, and only subsequently increase exposure.

5. Experiments and Results

5.1 Offline Evaluation vs. Baselines

The offline protocol applies very strict temporal partitioning reflecting production-causality and with no leakage to look-ahead: T-28-T-14 training, T-14-T-7 verification, and T-7-T testing. Features are derived only based on evidence available

by each split, and labels are only predicted out of past evidence (e.g., confirmed incidents, honeypot events, analyst adjudications). Evaluation is done micro-averaged over requests (representing the overall traffic) and macro-averaged over endpoints to eliminate domination by large-volume paths. The scoring code is affixed to the use commit that the model was trained on; the inference performs a different number of iterations with all cache warm and CPU pinned to mimic the single-request handler and gateway of the scoring code (***Raju, 2017)***.

The primary metrics are PR-AUC and ROC-AUC as ranking scores; recall@FBR or F1@FBR, with a fixed false-block-rate ceiling; and added milliseconds (added-ms) as on-path latency. In the case of thresholded metrics, a per-endpoint decision threshold is chosen on the validation window, meeting the FBR cap and yielding maximized F1; the same threshold is then used on the test window. Added-ms includes the replay of the test set on production-class nodes over the score range; a report is produced containing p50/p95/p99 inference, extraction time, and overall minor overhead.

Table 2: A summary of offline evaluation splits, metrics, latency, baselines, and tests

Component	Method/Setting	Key Parameters	Outcome
Temporal partitioning	Strict time-based splits	Train: T−28→T−14; Val: T−14→T−7; Test: T−7→T	Preserves causality; prevents look-ahead leakage.
Feature/label causality	Split-aware derivation	Features only from within split; labels from past incidents/honeypots/analyst calls	Realistic offline evaluation.
Metrics & thresholds	Ranking + enforcement metrics	PR-AUC, ROC-AUC; recall@FBR, F1@FBR; per-endpoint thresholds picked on Val, fixed on Test	Discrimination plus fixed-FBR operating points.
Latency measurement	Added-ms on production nodes	p50/p95/p99 for inference, feature extraction, total	Quantifies on-path cost.
Averaging scheme	Micro + Macro	Micro over requests; Macro over endpoints	Prevents domination by high-volume paths.

Component	Method/Setting	Key Parameters	Outcome
Baselines & statistics	Rules-only; Rules+Reputation; Per-endpoint rate-limit	10k stratified bootstrap CIs; DeLong (ROC-AUC); permutation (PR-AUC, recall@FBR)	Comparable baselines with robust significance.

Three baselines are involved in the comparison of their baselines. Rules-only runs the existing WAF and static rate-limit policy. Rules + R Rules are supplemented with coarse ASN/IP reputation buckets and allowlists. Per-endpoint rate-limit tokens a token-bucket to the distribution of a legitimate traffic type, per path_template, and labels sustained or otherwise excessive exceedance; per-path-template tokens are tuned such that per-path-template sustained proportionate p99 legitimate requests fit within budget. Confidence intervals on PR-AUC, recall@FBR, and the added-ms are based on 10,000 stratified bootstrap resamples; we use DeLonge's test to compare ROC-AUCs, and a permutation test to compare PR-AUCs and recall@FBR. Offline package also records calibration curves and precision-recall at operational thresholds to be directly compared to online stages.

5.2 Ablation Studies

The ablations measure which inputs and which modeling options are the most significant under a constrained on-path latency budget. The grouping of the behavioral counters is divided in (i) sliding-window request rates, (ii) unique paths per 60s, (iii) error-burst indicators, (iv) reputation and network context, (v) ASN, coarse geography, and network prevalence, and (vi) endpoint semantics, (v) HTTP method, (vi) path_template trie match, (vii) parameter entropy, (viii) token presence and length. Each subgroup is iteratively removed and retrained with fixed hyperparameters transferred down from the entire model and using the same temporal splits. A 95% bootstrap confidence interval is reported for 2PR-AUC, 2recall@FBR=0.5%, and 2added-ms.

Prioritization of permutation importance in each group identifies the few features that contribute most lift; those are candidates to retain when PARING the feature budget. Model-family swaps are performed under balanced feature set and latency costs: a calibrated L2-regularized logistic regression, a depth-limited gradient-boosted tree ensemble, and a compact one-hidden-layer MLP quantized to INT8. Distillation (tree→linear) is tested to preserve as much accuracy as possible at a reduced footprint. The effects of calibration are isolated, and the depletion of per-endpoint thresholds relative to a global threshold is employed to measure the improvement of risk tuning at the endpoint (***Whitehead,. 2023)***. This general trend-unequal signals that collectively outperform any one system-is in line with what is

known about multimodal modeling: when modalities are complementary, they can increase separability under noise and imbalances (***Singh, 2022)***.

5.3 Generalization & Robustness

Three settings of transfer measure generalization under each setting of transfer. Cross-region models trained in Regions A/B are evaluated in Region C, which has different ASN mixes, the prevalence of carrier-NATs, and diurnal patterns. They are thresholds that must remain constant to measure real transfer; the success standards must be a recall @FBR degradation less than 10% absolute, and its added-ms constant within + /- 0.5 ms of home regions. Cross-service in a case where a gateway is involved with multiple business units, a model trained on a high-risk service (login, payment endpoints) is applied on lower-risk services (search, catalog) with different error/volume characteristics; when recall@FBR stays within pre-specified boundaries without breaching latency limits, then acceptable transfer is achieved. Cold-start endpoints that were not observed in training are estimated based solely on global features and reputation and are then re-scored after 24 h-72 h during ongoing features warming to quantify bootstrapping speed.

Robustness is measured on oracle edge adversarial perturbation via reordered headers, addition of parameters of no consequence, randomized case, and rate-limited, bursty traffic meant not to exceed naive per-client rate limits. Each perturbation results in a report of worst-case recall@FBR, and a reliability diagram that captures calibration drift. Sensitivity analyses scale session window sizes (5, 10, 30 minutes) and tokenize path parameters using other strategies (character-level, word-level, and hashed subtoken bins). Resilience to sparse feature drop (e.g., missing ASN because of lookup timeouts) and to counter staleness by invalidating caches to simulate failover events is also measured (***Nambiar, 2021)***.

5.4 Online A/B, Shadow, and Canary

Before enforcement, a shadow phase marks 1 per cent of live traffic during seven days without having any impact on user experience. The shadow monitoring ensures that distributions of scores are stable across regions and hours via two-sample Kolmogorov-Smirnov tests and chi-squared tests of action-intent disagreement against rules-only baselines. Alarms activate when either the boundaries of the risk-deciles shift above or below a level of one decile or when the disagree-level of two consecutive hours shifts above 5 per cent. Decision thresholds are set to hold until shadow exit, and a canary phase is used to allow enforcement to apply to 0.1-1% of traffic per endpoint, curtailed by SLOs: added p95 <=5 ms, scorer timeout rate <=0.5%, no elevated 5xx. Canary guardrails should include automatic rollback on burn-rate, circle-breakers on scorer timeouts, and fail-open on low-risk endpoints.

Business impact is measured in three ways. The malicious acceptance rate (MAR) is the proportion of confirmed-malicious requests that succeed; expected to decrease markedly in the face of control, with Wilson-interval confidence bounds not overlapping. Mitigated fraudulent QPS is computed based on verified categorizations and honeypot validation, and broken down by tenant to identify disproportionate mitigation (***Liu, 2023)***. Difference-in-differences with tenant and region fixed effects may be used to monitor user conversion and abandonment deltas; a negative lift exceeding a pre-registered threshold results in rollback and threshold relaxation. Shadow/canary orchestration, policy bundle signing, and threshold promotion are added to the CI/CD pipeline in such a way that enforcement updates are versioned, auditable, and automatically tested with security tests- aligning the experimental workflow with DevSecOps best practices on advanced verification releases (***Konneru, 2021)***.

5.5 Error & Cost Analysis

Error analysis sits on actionable failure modes. False positives are grouped by endpoint, ASN, and user-agent family to identify systematic friction (such as particular mobile SDKs used behind carrier NATs). In each cluster, the least amount of evidence at decision time is shown to analysts, including windowed counts, counts of entropy, and course reputation bins; raw tokens and indicators are never presented. Counterfactual scoring interprets whether a slight shift of the threshold or removal of one vulnerable feature would have overturned the decision; remediations that are judged acceptable are made part of policy exceptions or constraints in features. False negatives are chased to blind spots, like lack of burst properties, improperly set thresholds on new endpoints, or adversarial traffic that masquerades as legitimate traffic; solutions can include adding countermeasures to overcome behavior transparent properties (NACs), narrower thresholds per endpoint, or low-cost secondary challenges on activating identities.

Cost is reported as per one million requests. The main one is CPU: CPU cost = (mean inference ms/request / 1000) x vCPU-hour price x 1,000,000. The memory headroom is credited by allocating a fractional node charge on the resident set and feature-cache size of the scorer. Decision-cache hit ratio is measured to estimate idle compute time, and idempotent GETs are tuned to have the cache TTLs. The autoscaling behavior is also profiled with synthetic burst workloads to ensure horizontal auto scaling scales without thrashing--scale-out and scale-in cool-down periods are also subjected to stress tests to ensure there is no oscillation behavior. The section ends with a net-benefit table comparing infrastructure cost with estimated fraud/abuse prevented to see where stricter policies are economically preferred and where log-only or challenge-first modes are preferred.

6. Discussion

6.1 Precision–Latency–Cost Trade-offs

The matter of securing operation points within an API gateway needs an overt balancing of latency, precision, and cost. Precision is modeled as an allowed false block rate (FBR) per endpoint; latency is the sum of time added by the feature extraction, scoring, and policy evaluation; cost is the additional CPU and memory used per request. A defensible policy starts with a utility function such as $U = v_{tp} \cdot TP - v_{fp} \cdot FP - c_{ms} \cdot \Delta latency_{ms} - c_{\$} \cdot cost\_per\_request$. Business tolerance is encoded into the coefficients. On a login endpoint, vtp will be large, as preventing account takeover is the critical factor; on a catalog endpoint, vfp will be large because it is detrimental to conversion to block browsing. The utility is determined by the risk threshold that meets the SLOs at the minimum cost, which is not necessarily the PR-AUC maximized one. This framing requires per-endpoint operating points and complex rules against one-size-fits-all thresholds.

Table 3: Precision–Latency–Cost trade-offs and operational budgets at the API gateway

Parameter	Value	Units/Domain	Notes
Latency budget: feature extraction	≤1.5	ms (p95)	Constant-time (O(1)) features; avoid heap allocations.
Latency budget: scoring	≤2.0	ms (p95)	CPU-only inference; single-record batches to improve tail latency.
Latency budget: policy evaluation	≤0.5	ms (p95)	Deterministic rule evaluation; minimal branching.
Latency budget: ML/policy subtotal	≤4.0	ms (p95)	Remaining headroom reserved for TLS and queueing.
Feature budget per request	≈50	features	O(1) computations; reuse parsed artifacts (e.g., path trie hits).
Inference batch size	1	records	Batch=1 reduces p95/p99; promotes determinism.
Sliding-	In-memory ring buffers	data structure	Keyed by client_ip_hash,

Parameter	Value	Units/Domain	Notes
window counters			ua_hash, tenant_id.
Queues & threading	Lock-free + NUMA pinning	implementation	Ensures determinism and low tail latency under load.
Decision cache (idempotent GETs)	Enabled	cache	TTL slightly larger than default to amortize repeated requests.
Utility function	$U = v\_tp \cdot TP - v\_fp \cdot FP - c\_ms \cdot \Delta latency\_ms - c\_\$ \cdot cost\_per\_request$	objective	Coefficients encode business tolerance per endpoint.
Endpoint weights: login	v_tp: High; v_fp: Medium	relative weights	Closed-leaning posture to prevent account takeover.
Endpoint weights: catalog	v_tp: Low; v_fp: High	relative weights	Open-leaning posture to preserve browsing/conversion.
Risk thresholds → actions	$t1 < t2 < t3$	ordering	ALLOW (risk < t1), RATE_LIMIT (t1–t2), CHALLENGE (t2–t3), BLOCK (≥ t3); hysteresis.
Rate limiting control	Multiplicative scaling	token bucket	Burst and sustain parameters scaled by risk score.
Challenge anti-relay	Opaque nonce	binding	Nonce bound to (client, endpoint, time).
Cost controls	Per-feature compute caps	cost	Limit CPU/memory per request; reuse parsed artifacts.

As highlighted in the Table 3 and Figure 5, the buffer of latency must be broken down into per-hop: feature extraction (≤1.5 ms), scoring (≤2.0 ms), and policy evaluation (≤0.5 ms), and the remnant must be available to TLS and queues. To maintain these budgets at high throughput, the scorer needs to limit feature counts (≈

50) to O(1) operations (as shown in Figure 6), construct sliding-window counters in memory using ring buffers keyed by client_ip_hash, ua_hash, and tenant_id, and must avoid heap allocations on the hot path. CPU-only inference with single-record batches improves tail latency; determinism under load is ensured by use of lock-free queues and NUMA pinning. Cost is regulated by limiting the amount of per-feature compute, reusing previously-parsed artifacts (such as path trie hits), and caching decisions to idempotent GETs with slightly larger TTL to spread the amortized cost across repeated requests.

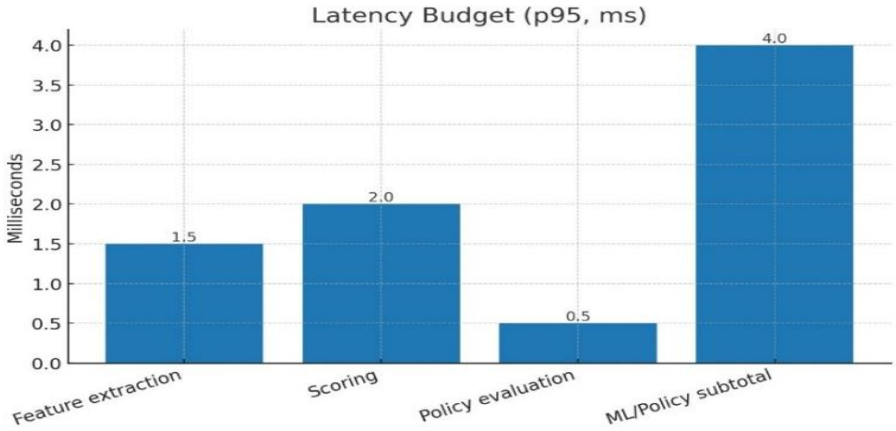

Figure 5: p95 latency budget by pipeline stage (feature extraction, scoring, policy)

Rate limiting, CAPTCHA, or step-up authentication are explicit steps in the same decision curve as BLOCK and are not add-on fallback mechanisms. A pragmatic mapping is risk < t1 → ALLOW; t1–t2 → RATE_LIMIT; t2–t3 → CHALLENGE; ≥ t3 → BLOCK. There are specific user-experience costs/recovery paths per-action; thus, per-action curves should reflect an understanding of business value, and the boundaries between congested and uncongested paths should be hysteretic to prevent flapping. In CHALLENGE, a nonce, which is opaque and bound to the client, endpoint, and time, is used to prevent relay (***Sharma; Jain, & Chandavarkar, 2020)***. To increase the risk score and control the volume without risking the backend, the burst and sustain of token-bucket settings on RATE_LIMIT are multiplicative. On login, the sequence leans to closed, and on catalog, it leans less to closed, leading to open, with only telemetry sampling when there is a marginal risk.

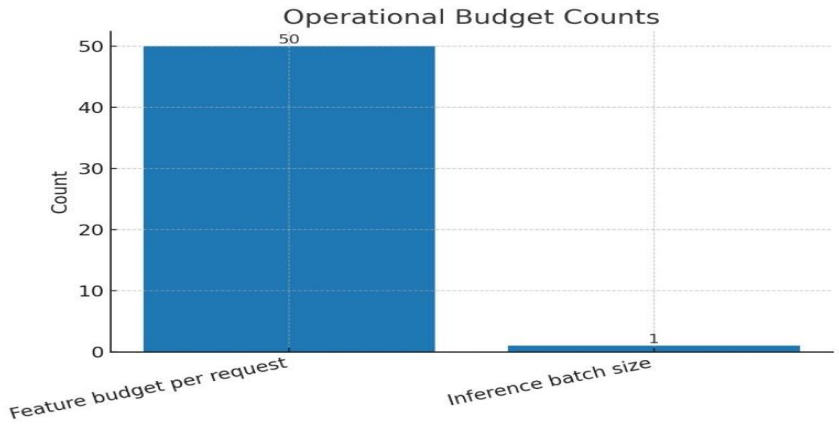

Figure 6: Operational budgets: feature count per request and inference batch size

6.2 Reliability, Drift, and MLOps

Score and policy engine scoring and policy engines are considered SLO-bearing services. Timeouts and misclassifications must be defined as erroneous budgets, and circuit breakers must switch to deterministic rules-only behavior in case the rate of timeouts crosses a burn threshold. Shadow evaluation, triggered initially by a small amount of traffic and followed by a small-traffic canary with automatic rollback contingent on guardrails-surge in 4xx at high-value endpoints, p95 latency regression, or the shift in score deciles, is required by Canary etiquette to certify calibration and stability of scores across traffic. Observability should have red/black metrics (latency, throughput, error), score histograms, counts of decisions by action, and per-endpoint FBR estimates calculated by analyst adjudications.

Model and feature drift will take place. The execution of the Population Stability Index (PSI) and Kolmogorov-Smirnov (KS) tests should be ongoing with respect to key features like request_rate_1m, unique_paths_10m, and param_entropy, as well as the final score. Alarms should be tiered: “yellow” when PSI $\geq$ 0.1, “red” when PSI $\geq$ 0.25 for multiple consecutive windows; parallel criteria apply to KS p-values and to decile-level calibration error. Retraining cadence should be data-driven: in case the drift continues over a week or the recall@FBR falls beyond a certain deterioration tolerance in the shadow data, trigger a retraining, using a timely separated validation dataset to prevent leakage. Deployment must be blue-green signed artifacts, versioned feature schemas, and reversible migrations of policy bundles.

On the multi-region scale, problems with consistency occur in online counters and feature stores. Sliding windows calculated per PoP must stay local to facilitate on-path decisions, to avoid lump volume latency; only aggregated analytics need final reconciliation. When a document-oriented store backs asynchronous analytics or feedback queues, the choice of write concern, read concern, and session guarantees must consider both speed and correctness considerations. The majority writes with causal reads can often afford a reliability envelope that will exclude stale or phantom

updates of security-critical signals when heavily loaded (***Dharsee, 2023)***. Conversely, TTLs are carefully bounded, or we may purge obsolete windows. This pattern has low on-path latency, and on- and offline evaluation remain reproducible and auditable.

6.3 Privacy, Governance, and Compliance

Minimization and auditability are at the basis of the privacy posture of gateway-centric prevention. Payload bodies should be dropped unless a deep-packet inspection path is expressly allowed; IP, user-agent, and email must be salted hashes rotated at regular intervals; ASN or geo can also be stored at a coarse resolution. The stores should also impose column-level access regulations to prevent analysts from accessing the raw identifiers, but only risk scores and aggregated counters (***Ferrari, 2022)***. Retention policies must distinguish the hot path (seconds to hours), training data (days to weeks), and audit logs (months), with cryptographic hash-chaining used on decisions (policy_version, rule_id, action, evidence_digest) to ensure tamper evidence. Governance must presuppose post-deployment validation that policies are compiled, meet guardrail tests, and result in passing regression suites referencing known incident playbooks.

Regional constraints complicate deployment. These make it more challenging to deploy Features that are legally available in some jurisdictions and not others (like device fingerprinting or IP-based reputation) ***(Arshad; Jantan, & Omolara, 2019)***. As a result, these features must rely on a region-capability matrix created during the build process and enforced by the policy engine at run time. The same model may not be non-portable because feature distributions or legal allowances may vary; teams are advised to adhere to region-specific calibration tables and, if needed, region-specific models trained on only compliant fields. A shift-left approach furthers this position by shifting security and compliance validation to earlier phases of production delivery, thereby reducing the cost of failure and allowing policy drift to be identified earlier in the process before a production rollout. The controls required to export data in analytics pipelines should ensure that the training data does not violate sovereignty rules and that no aggregation across jurisdictions of raw data is conducted aside from privacy-preserving summaries.

The most crucial bypass vectors that should be focused on during a red-team exercise are model-aware. Probable countermeasures involve reordering transmission heads, padding parameters with random data to confound entropy analysis, conducting low and slow scans to avoid burst triggers, and using residential proxies to escape naive reputation-based access denials. To increase attacker overheads, the system may randomly shuffle a low proportion of the challenges, canary instances in honeypot endpoints, and evaluate coverage of test points in feature space to focus adversarial attack testing. Governance must cover red-team reviews of policy diffs regularly, require open-book drills training against anticipated credential-stuffing attacks, and table-top activities to ensure that rollbacks are timely, communications are appropriately structured, and tools are available to analysts.

6.4 Limitations & Threats to Validity

Limitations notwithstanding, careful engineering was done to avoid them. Label noise is pervasive as many positives will be identified by heuristics or not known to analyst (or long enough they are not confirmed); training ovesamples heuristic proxies or correlated enforcement rather than malicious actors; label noise of undetermined affirmation that may be captured by the heuristic rules or a delayed affirmation-based heuristic may remain present, this is because the heuristics rules may capture such instances or may be reflected by delayed analyst confirmation of assignments. Weaker-label modeling, label smoothing, cross-endpoint consistency checks, and semi-supervised augmentation using high-confidence anomalies are applicable mitigations, but residual bias is likely. Sampling bias is inevitable; training datasets include disproportionately popular endpoints and business hours, which reduces recall on long-tail endpoints and off-peak geographies. This effect (though not wholly) is mitigated by targeted oversampling, per-endpoint thresholds, and periodic "tail sweeps" during evaluation.

Externality faces a threat to external validity because of non-stationarity. Changes in seasonal traffic, a product launch, or a new bot structure can cause changes in distribution patterns that are faster than retraining cycles can adapt. Frequent refreshing of calibration, dynamic per-endpoint thresholds, and guardrail rules can slow degradation. However, they cannot prevent it in the event of upstream changes to semantics, e.g., a new login flow that changes token-error distributions (***Prasad, 2019)***. Leakage of Proxy-signals is always a threat: features like waf_blocked and challenge_failed, which offer helpful conveniences, can result in a feedback loop effect that skews apparent accuracy. Such fields ought not be used as training features, or used with just monotonic constraints and adversarial tests that subject their availability to perturbations.

There are latency budgets that limit the richness of model classes and features, and thus limit the accuracy. The additional headroom afforded by two-stage designs and quantization can be lost to stealthy low-and-slow attackers that masquerade as legitimate traffic, and/or are difficult to detect without deeper session and graph context that is also prohibitively costly to compute on the hot path in the gateway. Evaluation can be misleading: offline PR-AUC does not consider user-experience costs, whereas online A/B tests may be plagued by interference effects when submitted adversaries modify their guesses during the experiment. Teams should triangulate decisions on offline metrics (fixed FBR), short canaries with stricter rollback, and long-running shadow tests, which monitor calibration and drift. There remains doubt; policies should be designed to allow rapid reversal in the short and medium term, and enforcement levels must default to challenge or rate-limit where confidence in beliefs is limited yet not wholly absent, whilst retaining sufficient evidence to support post-incident learning.

7. Future Consideration

7.1 Semi-/Unsupervised & Active Learning

Further versions can capitalize on unlabeled gateway telemetry to identify emerging abuse with less brittle signatures. These self-supervised goals, masked-header prediction, path-template denoising, and contrastive learning across sessions on request streams can be used to learn compact embeddings that can distinguish between abnormal traffic and typical workloads. Such embeddings can be used to feed lighter anomaly scorers like Isolation Forest to identify zero-day behavior and bootstrap pseudo-labels that lower manual curation.

The practical loop batches the minutes of traffic, recomputes embeddings offline, and sends a small "anomaly model" and lookup tables to the edge. Adopting uncertainty sampling will focus analyst attention to the high-entropy or margin-close cases, whereas coverage constraints will guarantee that all endpoints with low volumes are still considered. The loop does not associate enforcement outcomes with the learning signal to prevent leakage of the learning by an opponent ***Hu; Zhang; Nakamura; Bajcsy, & Fisac, 2023)***. To the extent that self-supervised learning can provide meaningful improvements on perception tasks via unlabeled datasets, similar gains can be expected when it comes to log embeddings (***Singh, 2023)***.

7.2 Graph & Sequence Signals

Critical gestures often have a coordinating character as opposed to individual demands. A dynamic heterogeneous graph with nodes {client_ip_hash, account, device_fingerprint, token_id, ASN, endpoint, tenant} and time-stamped edges {login, request_to, token_failure} enables features such as degree bursts, temporal PageRank, community detection, and label propagation from confirmed incidents. Temporal motif counts many accounts, many endpoints in seconds, or many accounts one IP within minutes, and is amenable to counting using sketches (count-min, HyperLogLog) and compact windowed counters, which can be streamed in O(1) (***Chothia, 2020)***.

As shown in the figure below, an incremental heterogeneous graph pipeline consumed within a gateway captures events to generate nodes { client_ip_hash, account, device fingerprint, token id, ASN, endpoint, tenant } and edges { login, request to, token failure } that are time-stamped. Degree bursts, temporal PageRank, community detection, and label propagation are also computed, and temporal motif counts are streamed along with count-min/HyperLogLog sketches and windowed counters to enable coordinated-abuse exposure, such as per-endpoint risk aggregation scores.

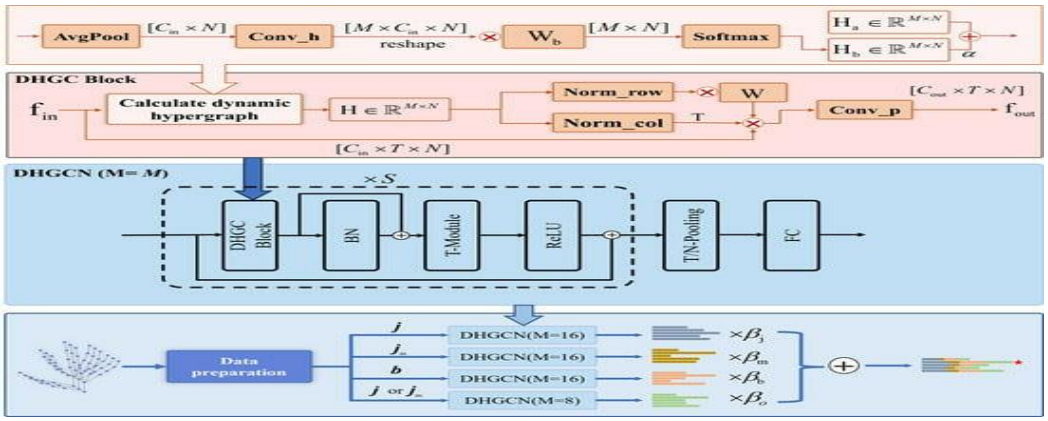

Figure 7: Dynamic heterogeneous graph pipeline for temporal motifs and coordinated threat detection

Short-horizon sequence models (GRU or temporal CNN) over per-entity request lines can achieve cadence, burstiness, and order statistics; short weights/sequences should be linearized using a surrogate, or compiled in WASM with INT8 weights. Heavy graph updates are run out-of-band, so the gateway consumes precomputed scores and the small neighborhood cache that has expired with an ETag.

7.3 Federated/Privacy-Preserving ML

Training across regions and tenants should happen where data resides, and in-aggregate updates should be shared. A pragmatic design educates per-region teachers on compliant feature subsets, safely aggregates those into a global student, and distills them into a quantized gateway model. The sensitivity of dimensions (device fingerprints, organization identifiers) is budgeted on a differential-privacy basis (reported on the model card).

Federated rounds involve vigorous sampling of clients, and checks on held-out regional cube segments, together with drifting alerts when the gradient norms or score histograms diverge—global rollback. The international student is shadowed at small traffic slices before promotion, and regional fallbacks are maintained for rollback (***Coats, 2019)***. The goal is to make the generalization over tenants more effective, but preserve the data-minimization warranties and consistent enforcement on the edge. Trusted signature systems using hardware-backed secure enclaves can be used to back up model bundle signature keys.

7.4 Policy Optimization & Human Factors

Contingent risk scores are valuable only when translated into actions that reduce harm with latency and cost constraints. A constrained contextual bandit determines {allow, rate-limit, challenge, reauth, block} based on rewards that are a mixture of malicious-acceptance reduction, user friction, and compute cost; the set of constraints encodes endpoint SLOs and regulatory limits. Offline counterfactual

assessment using inverse propensity scoring constrains risk with bounds. Online, guardrails limit the rate of challenge, impose a minimum recall per endpoint, and raise burn-rate alerts. Analyst tooling ought to display the explanations (top features and exemplars), record override as counterfactual feedback, and show the policy-diffs during canaries. Budget and throughput constraints necessitate cost-aware action policies; security improvements that do not consider the constraint create a bottleneck in one place or increase platform spend (***Chavan, 2023)***.

8. Conclusions

This work shows that sustained high-volume, on-path prevention can be achieved when the API gateway is viewed as a latency-sensitive inference and rule decision stack. Prevention is effected as synchronous actions- ALLOW, RATE_LIMIT, CHALLENGE, REAUTH, BLOCK, SANDBOX mapped against calibrated risk scores based on constant time features and a sliding-window counter. There are apparent engineering limitations, too: a per-request feature limit of around fifty or so, a ring-buffer state mechanism keyed using hashes of client and tenant identifiers, and scorers that run in-process (as in the case of WebAssembly), or behind a gRPC interface with 2-4 ms latency limits and circuit-breaker safeguards. Decision quality relies on per-endpoint calibration and thresholds, hysteresis to prevent flapping, and a deterministic policy DSL that evaluates a directed acyclic rule graph with short-circuit guardrails such as allowlists and mTLS requirements.

Reliability is provided by bounded queues, backpressure, NUMA-pinned scorer threads, and fail-open or challenge-first according to endpoint risk. Multi-region, multi-tenant governance is achieved via bundles that are signed and hot-reloaded, tenant-scoped counters, and optional CRDT-style reconciliation of analytics. However, on-path enforcement is still locally enforced. Evaluation uses strict time-division without leakage, reports PR-AUC, recall at fixed FBR, and additional milliseconds, and validates online using shadow and canary phases that report malicious-acceptance reduction without SLO regressions. The entire body of evidence shows that a combination of behaviorally informed privacy-preserving telemetry (hashed identifiers, path templates, coarse ASN) and small, calibrated models and deterministic policy can have an effect that can be measured at the internet scale.

The results can be summarized into a deployable playbook. Per-hop budgets (feature $\leq$1.5 ms, scoring $\leq$2.0 ms, policy $\leq$0.5 ms) are imposed in the current line profiling mode in cache-cold and bursty conditions. Implement on-path lifetime features on a path-trie feature store with ring buffers and LFU caches; expose only O(1) primitives like the presence of a header, path-trie hits, token statistics, short-window request and failure rates, rarity, and entropy. Prefer a calibrated linear or depth-limited tree model, reduce to a small surrogate model, and quantize when convenient; deploy as an isotonic lookup that maps raw scores to risk scores with monotonic safety-critical requirements imposed on the inputs. Determine per-endpoint thresholding by utility function weighting business value and user friction; favor

challenge-first or fail-closed to high-risk endpoints and fail-open no-risk reads. Bounded queues, backpressure, rule-enhanced lock-free counters, NUMA pinning, and circuit breakers that degrade to rules-only mode on scorer timeouts or error-budget burn.

Instrument red/ black metrics, score histograms, decile drift monitors, decision distributions, and added-ms by hop; trigger burn-rate alerts and keep rollback one flag away. Gradually roll out through shadow scoring over a week and canary enforcement with 0.1-1% of traffic, with automatic rollback once guardrails have been tripped; pair decision caching of idempotent GETs with signed audit logs and SIEM export. Define budgets specifically in terms of CPU-hours per one million scored requests, memory headroom, and cache hit rates, and contrast against the estimate of fraud and abuse prevented to provide action curves. Reproducibility under information constraints must include ship schemas, preprocessing graphs, feature dictionaries, calibration tables, examples, replayers, and synthetic generators that resemble burstiness and session-structure, notebooks to perform temporal splits, and guardrails to rollout.

References;

Arshad, H., Jantan, A., & Omolara, E. (2019). Evidence collection and forensics on social networks: Research challenges and directions. *Digital Investigation*, *28*, 126-138.

Balamurugan, H. V. (2023). *Enhancing Individual Privacy Preservation in Multi-Tenancy Cloud Environments through Secure Multi-Party Computations: A Differential Privacy-Based Data Partitioning Strategy* (Doctoral dissertation, Dublin, National College of Ireland).

Chavan, A. (2023). Managing scalability and cost in microservices architecture: Balancing infinite scalability with financial constraints. Journal of Artificial Intelligence & Cloud Computing, 2, E264. http://doi.org/10.47363/JAICC/2023(2)E264

Chiariotti, F., Kucera, S., Zanella, A., & Claussen, H. (2019). Analysis and design of a latency control protocol for multi-path data delivery with pre-defined QoS guarantees. *IEEE/ACM Transactions On Networking*, *27*(3), 1165-1178.

Chothia, Z. (2020). Explaining, Measuring and Predicting Effects in Layered Data Architectures (Doctoral dissertation, ETH Zurich).

Coats, C. C. (2019). *Reducing memory persistency overheads with transparent out-of-place updates* (Doctoral dissertation, University of Illinois at Urbana-Champaign).

Dharsee, K. (2023). *Critical Hardware Towards Software Security Enforcement*. University of Rochester.

Fainchtein, R. A. (2023). *No Sieve is Good Enough: Examining the Creation, Enforcement and Evasion of Geofilters* (Doctoral dissertation, Georgetown University).

Fajana, O. (2023). Novel Techniques for Detecting Tor Botnets.

Ferrari, E. (2022). *Access Control in Data Management Systems: A Visual Querying Perspective*. Springer Nature.

Hu, H., Zhang, Z., Nakamura, K., Bajcsy, A., & Fisac, J. F. (2023). Deception game: Closing the safety-learning loop in interactive robot autonomy. *arXiv preprint arXiv:2309.01267*.

Konneru, N. M. K. (2021). Integrating security into CI/CD pipelines: A DevSecOps approach with SAST, DAST, and SCA tools. *International Journal of Science and Research Archive*. Retrieved from https://ijsra.net/content/role-notification-scheduling-improving-patient

Kumar, A. (2019). The convergence of predictive analytics in driving business intelligence and enhancing DevOps efficiency. International Journal of Computational Engineering and Management, 6(6), 118-142. Retrieved from https://ijcem.in/wp-content/uploads/THE-CONVERGENCE-OF-PREDICTIVE-ANALYTICS-IN-DRIVING-BUSINESS-INTELLIGENCE-AND-ENHANCING-DEVOPS-EFFICIENCY.pdf

Larsson, L., Tärneberg, W., Klein, C., Kihl, M., & Elmroth, E. (2021). Towards soft circuit breaking in service meshes via application-agnostic caching. *arXiv preprint arXiv:2104.02463*.

Liu, G. (2023). Investigating Security Threats of Resource Mismanagement in Networked Systems.

Mei, G., Pan, L., & Liu, S. (2022). Heterogeneous graph embedding by aggregating meta-path and meta-structure through attention mechanism. *Neurocomputing*, *468*, 276-285.

Nambiar, N. (2021). Attack resilience of cache replacement policies: Implementation and experimentation in SDN.

Nardini, G., Sabella, D., Stea, G., Thakkar, P., & Virdis, A. (2020). Simu5g–an omnet++ library for end-to-end performance evaluation of 5g networks. *IEEE Access*, *8*, 181176-181191.

Nyati, S. (2018). Transforming telematics in fleet management: Innovations in asset tracking, efficiency, and communication. International Journal of Science and Research (IJSR), 7(10), 1804-1810. Retrieved from https://www.ijsr.net/getabstract.php?paperid=SR24203184230

Paracha, M. T. (2023). *Measurement Techniques to Understand How Diversity in TLS Implementations & Deployments Influences Protocol Security* (Doctoral dissertation, Northeastern University).

Prasad, A. (2019). *Structured Information Extraction for Scientific Documents* (Doctoral dissertation, National University of Singapore (Singapore)).

Qiu, H., Banerjee, S. S., Jha, S., Kalbarczyk, Z. T., & Iyer, R. K. (2020). {FIRM}: An intelligent fine-grained resource management framework for {SLO-Oriented} microservices. In *14th USENIX symposium on operating systems design and implementation (OSDI 20)* (pp. 805-825).

Raju, R. K. (2017). Dynamic memory inference network for natural language inference. International Journal of Science and Research (IJSR), 6(2). https://www.ijsr.net/archive/v6i2/SR24926091431.pdf

Repanovici, R. M., Nedelcu, Ș., Tarbă, L. A., & Busuioceanu, S. (2022). Improvement of emergency situation management through an integrated system using mobile alerts. *Sustainability*, *14*(24), 16424.

Sardana, J. (2022). Scalable systems for healthcare communication: A design perspective. *International Journal of Science and Research Archive*. https://doi.org/10.30574/ijsra.2022.7.2.0253

Shaked, A., Cherdantseva, Y., Burnap, P., & Maynard, P. (2023). Operations-informed incident response playbooks. *Computers & Security*, *134*, 103454.

Sharma, S., Jain, S., & Chandavarkar, B. R. (2020, October). Nonce: Life cycle, issues and challenges in cryptography. In *ICCCE 2020: Proceedings of the 3rd International Conference on Communications and Cyber Physical Engineering* (pp. 183-195). Singapore: Springer Nature Singapore.

Singh, V. (2022). Multimodal deep learning: Integrating text, vision, and sensor data: Developing models that can process and understand multiple data modalities simultaneously. International Journal of Research in Information Technology and Computing. https://romanpub.com/ijaetv4-1-2022.php

Singh, V. (2023). Enhancing object detection with self-supervised learning: Improving object detection algorithms using unlabeled data through self-supervised techniques. International Journal of Advanced Engineering and Technology. https://romanpub.com/resources/Vol%205%20%2C%20No%201%20-%2023.pdf

Song, Y., Jiang, H., Zhang, H., Tian, Z., Zhang, W., & Wang, J. (2023). Boosting studies of multi-agent reinforcement learning on Google research football environment: The past, present, and future. *arXiv preprint arXiv:2309.12951*.

Starc, R. P. (2023). Exploring the Microarchitectural Implications of Serverless Workloads Using RISC-V (Master's thesis, ETH Zurich).

Whitehead, J. F. (2023). *Advanced X-ray Imaging Techniques for Hepatic Arterial Blood Velocity Measurement*. The University of Wisconsin-Madison.