# Risk-Aware Federated Graph-Temporal Learning for Cross-Domain Application Programming Interface Security Monitoring

**Xuhua Ai[1], Yiting Huang[1], Qi Meng[1], Zhaoli Chen[1], Yun Dong[1], Yuan Yin[1,*]**
[1]Digital and Intelligent Operations Center, Guangxi Power Grid Co., Ltd.., Nanning, China
Corresponding author: Yuan Yin (e-mail: 1014258769@qq.com)

**ABSTRACT** Application programming interfaces have become the principal communication surface of cloud-native and distributed business systems, but their security telemetry is fragmented across gateways, service meshes, application logs, and organizational domains. Centralized monitoring is difficult when request payloads, identity tokens, object identifiers, and business parameters cannot leave their original security boundaries. This paper presents FedAPI-Mon, a risk-aware federated graph-temporal framework for cross-domain application programming interface security monitoring. Each participant converts locally observed requests into windowed call graphs that jointly represent endpoint semantics, identity transitions, object access, response status, latency, and invocation order. A relational graph-attention encoder learns lateral dependencies among services, while a temporal self-attention encoder models multi-step behaviors that are weak when viewed as isolated requests. A risk-aware aggregation rule then weights local updates by threat coverage, update consistency, and validation reliability, reducing model drift under non-independent and non-identically distributed traffic. Finally, an adaptive boundary combines classification confidence, representation distance, and short-term baseline deviation to identify both known and previously withheld attack patterns. Controlled replay experiments containing 1.24 million requests, 40 monitoring items, 12 services, and six attack families show a macro-F1 of 96.4%, a receiver-operating-characteristic area of 98.3%, and a 1.7% false-positive rate. In leave-one-attack-family-out tests, the method retains a macro-F1 of 91.8%. It also compresses 96.2% of repetitive alerts while preserving 95.4% of malicious events. The results indicate that federated graph-temporal learning can improve API risk visibility without centralizing sensitive request data.

**INDEX TERMS** Anomaly detection, API security, Federated learning, Graph neural network, Risk monitoring, Temporal modeling

## I. INTRODUCTION

Application programming interfaces (APIs) connect mobile applications, cloud services, edge devices, data platforms, and automated operation tools. Their role has expanded from a narrow software integration mechanism into an exposed business control surface. A single API request may carry an identity token, an object identifier, a resource action, a business parameter set, and a chain of downstream calls. Consequently, security failures are not limited to malformed packets. They also include broken object authorization, broken authentication, unrestricted resource consumption, sensitive business-flow abuse, server-side request forgery, and unsafe consumption of third-party APIs [1]. These risks frequently arise from a request that is syntactically valid but inconsistent with the caller's identity, historical behavior, access path, or business context.

Conventional API monitoring usually relies on static rules at an API gateway, signatures in a web application firewall, or isolated anomaly scores derived from request rate and response status. Such controls remain useful, but they have three limitations. First, a gateway observes the north-south request but may not reconstruct the east-west service calls triggered by that request. Second, authorization abuse is relational: the risk depends on the combination of subject, object, endpoint, method, and invocation path rather than on one field alone. Third, business traffic varies substantially among regions, applications, tenants, and time periods. A global threshold trained on centralized samples therefore tends to over-alert on legitimate local behavior while under-detecting rare behavior that appears in only one domain.

Federated learning (FL) offers a way to train a shared detector without transferring raw telemetry to a central repository [2]. However, directly applying federated averaging to API monitoring is insufficient. API observations

are strongly non-independent and non-identically distributed (non-IID): one participant may mainly process identity services, another may host asset queries, and another may expose write-intensive control functions. Local label coverage is also uneven because some domains have never observed particular attack families. Simple sample-count weighting can consequently amplify a large but low-diversity client and suppress a small client carrying rare, security-relevant evidence. Moreover, an API incident often unfolds as a sequence of related calls. Flattening those calls into independent feature vectors discards the very path information needed to distinguish legitimate automation from reconnaissance, privilege traversal, and object-level abuse.

This paper develops FedAPI-Mon, a cross-domain monitoring framework that combines relational call-graph learning, temporal behavior modeling, risk-aware federated aggregation, and adaptive decision boundaries. Rather than transferring a generic distributed-learning pipeline unchanged, the proposed work redesigns the representation, objective function, aggregation evidence, decision rule, and evaluation protocol for API security. The principal contributions are as follows.

1) A windowed API call graph jointly represents endpoint, identity, object, status, latency, payload-schema, and call-order evidence while keeping raw request content local.
2) A graph-temporal encoder captures both lateral service dependencies and longitudinal multi-step behavior. A focal classification term is combined with a prototype-based contrastive boundary and a proximal consistency term to handle severe class imbalance and client drift.
3) A risk-aware aggregation mechanism assigns each participant a weight based on attack-family coverage, update consistency, and validation reliability instead of relying only on sample volume.
4) An adaptive monitoring boundary fuses supervised confidence, distance from the local normal prototype, and exponentially weighted baseline deviation. This design supports known-attack classification and previously withheld attack-family detection through one risk score.
5) Controlled replay experiments evaluate detection accuracy, unknown-attack generalization, non-IID robustness, alert compression, and runtime overhead. The experimental values are reported as controlled laboratory results rather than utility field-deployment claims.

## II. RELATED WORK

### A. API SECURITY MONITORING

The OWASP API Security project identifies authorization, authentication, resource consumption, business-flow, configuration, inventory, and unsafe downstream-consumption risks as recurring API security concerns [1]. Existing operational controls commonly combine endpoint inventories, access policies, rate limits, schema validation, gateway rules, and log analytics. These measures are effective for explicit policy violations, but subtle attacks may remain within valid syntax and acceptable instantaneous request rates. Examples include enumerating object identifiers at a human-like rate, alternating permitted and forbidden endpoints, or using a legitimate token to reach an abnormal service path.

Sequence-based anomaly detection has been applied to system and application logs. DeepLog models normal log-key sequences with a recurrent neural network and flags events that violate the learned sequence pattern [3]. Such methods establish the value of order information, but a linear log sequence does not explicitly express the many-to-many relations among identities, endpoints, services, and objects. API monitoring therefore benefits from a representation that retains both order and topology (Figure 1).

### B. GRAPH AND TEMPORAL REPRESENTATION LEARNING

Graph neural networks aggregate information from related nodes and are well suited to service-call, provenance, and entity-interaction structures. Graph attention networks learn neighbor-specific weights without requiring a fixed spectral operator [4]. For API monitoring, attention can distinguish a high-risk cross-service edge from routine repeated calls, while relation types can separate “invokes,” “accesses,” and “authenticated-as” semantics.

Self-attention models long-range dependencies by comparing all positions within a sequence [5]. It is useful when a suspicious request is separated from its causal precursor by several benign-looking calls. However, temporal attention alone treats each event as a token and may overlook the changing service topology. FedAPI-Mon therefore applies relational graph attention inside each time window and temporal attention across consecutive window representations.

### C. FEDERATED LEARNING UNDER HETEROGENEITY

FedAvg trains local models and aggregates their updates, reducing the need to move raw data [2]. FedProx adds a proximal term to limit deviation from the global model under system and statistical heterogeneity [6]. SCAFFOLD uses control variates to correct client drift [7]. These methods provide general distributed-optimization foundations, but they do not distinguish between a client rich in rare attack evidence and a client dominated by repetitive normal traffic. In API monitoring, aggregation should consider security evidence quality in addition to optimization stability. FedAPI-Mon introduces coverage, consistency, and reliability factors that are computed from non-sensitive summary statistics and validation outcomes.

## III. PROBLEM FORMULATION AND THREAT MODEL

### A. CROSS-DOMAIN MONITORING SETTING

Consider N API-monitoring domains. Domain i observes a private request stream $D_i=\{e_t^i\}$, where an event contains locally tokenized endpoint, method, subject role, object class, status code, latency, payload-schema digest, and trace

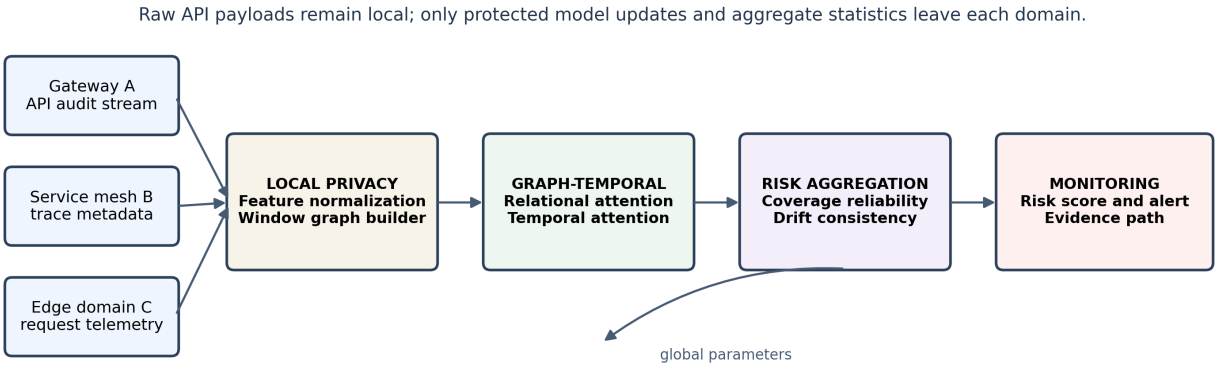


**Fig. 1.** Cross-domain API security monitoring architecture. Raw request payloads remain inside each local security boundary. The coordinator aggregates protected model updates and returns a global detector, while the monitoring layer produces risk scores, alerts, and evidence paths.

relation. Sensitive values such as raw tokens, object identifiers, request bodies, and business parameters never leave the domain. During federated round r, a subset $C_r$ trains from local graph windows and uploads a model update $\Delta\theta_i^r$ plus bounded aggregate statistics.

The optimization objective is

$$\min_{\theta} \; F(\theta) = \sum_{i=1}^{N} p_i F_i(\theta), \qquad F_i(\theta) = \mathbb{E}_{(G,y)\sim D_i}[\mathcal{L}(f_\theta(G), y)]\,. \tag{1}$$

where G is a graph-temporal sample, $y\in\{0,\ldots,K\}$ is a normal or known-attack label, and $p_i$ is not fixed solely by sample count but is learned from risk-aware evidence.

### B. ADVERSARY AND PRIVACY ASSUMPTIONS

The monitored adversary may use valid credentials, manipulate object identifiers, traverse rarely used endpoint paths, replay requests, induce resource exhaustion, probe versioned interfaces, or combine low-intensity actions across several minutes. The method does not assume that every attack contains an exploit signature. It instead seeks inconsistent relations and temporal deviations.

The coordinator is assumed honest-but-curious: it follows the aggregation protocol but should not receive raw traffic. Individual participants may have noisy labels or unrepresentative local distributions. Model poisoning and secure-aggregation cryptography are outside the main scope; robust update clipping is included, but a complete Byzantine-resilient protocol remains future work. The monitoring output is intended to prioritize investigation and does not autonomously block a production request.

## IV. FEDAPI-MON METHOD

### A. WINDOWED API CALL-GRAPH CONSTRUCTION

Events are partitioned into overlapping windows of W requests or $\Delta t$ seconds. A heterogeneous graph $G_t=(V_t,E_t,X_t,R_t)$ is built for each window. Nodes represent subjects, endpoint templates, service instances, object classes, and response groups. Edges represent invocation, authentication, object access, downstream propagation, and temporal succession.

For event e, the local feature vector is

$$x_e = [z_{ep}\|z_m\|z_r\|\phi_s\|\phi_b\|\phi_t] \tag{2}$$

where $z_{ep}$, $z_m$, and $z_{role}$ are learned embeddings. The behavior group contains normalized latency, request-rate deviation, object-access entropy, and payload-schema distance; the remaining scalar features describe response category and trace depth. Endpoint paths are normalized into templates and identifiers are salted-hashed locally, preventing literal values from entering the federation.

Object-access entropy is computed as

$$H_{obj}(u,t) = -\sum_{o\in\Omega_{u,t}} p_o \log p_o \tag{3}$$

where $\Omega_{u,t}$ is the set of object classes accessed by subject u in window t. A sudden increase can indicate enumeration, while a low-entropy repeated write pattern may indicate automated abuse.

### B. RELATIONAL GRAPH-ATTENTION ENCODER

For node v, relation-specific attention aggregates its neighbors:

$$h_v^{(\ell+1)} = \sigma\left(\sum_{q\in\mathcal{R}}\sum_{u\in N_q(v)} \alpha_{vu}^{q,\ell} W_q^{\ell} h_u^{(\ell)}\right) \tag{4}$$

with

$$e_{vu}^{q} = \mathrm{LeakyReLU}\left(a_q^{\top}[W_q h_v \| W_q h_u]\right), \qquad \alpha_{vu}^{q} = \mathrm{softmax}_{u\in N_q(v)}(e_{vu}^{q})\,. \tag{5}$$

The graph embedding $g_t$ is obtained through gated attention pooling. Because attention is relation-specific, a rare cross-service invocation can receive a higher weight than a frequent health-check edge even if the two requests share similar scalar features.

### C. TEMPORAL BEHAVIOR ENCODER

Consecutive graph embeddings $(g_{t-L+1},\ldots,g_t)$ are augmented with position and time-gap encodings. Multi-head temporal self-attention produces

$$A_t = QK^\top/\sqrt{d} + M_{time},\quad Z = \mathrm{softmax}(A_t)V. \tag{6}$$

where $M_{time}$ penalizes pairs separated by a time gap larger than the monitoring horizon. The final state $z_t$ summarizes the evolving call path. This representation distinguishes a legitimate bulk job with a stable scheduled pattern from a short-lived burst that traverses unexpected identities and endpoints.

### D. JOINT LOCAL OBJECTIVE

Normal requests dominate API telemetry, so the supervised component uses focal loss [8]–[10]:

$$\mathcal{L}_{cls} = -\sum_{c=0}^{K} \alpha_c (1-\hat{p}_c)^{\gamma} y_c \log \hat{p}_c \tag{7}$$

To improve open-set sensitivity, each client maintains a normal prototype $c_i$. The contrastive boundary loss pulls normal representations toward the prototype and pushes labeled attacks beyond margin m:

$$\mathcal{L}_{bnd} = I_0 d_i^2 + I_1 [m - d_i]_+^2 \tag{8}$$

Here, $d_i=\|z-c_i\|_2$, while $I_0$ and $I_1$ select normal and attack samples. The local objective at federated round r is

$$\mathcal{L}_i^r = \mathcal{L}_{cls} + \lambda_b \mathcal{L}_{bnd} + \mathcal{L}_{prox} + \lambda_s \mathcal{L}_{smooth} \tag{9}$$

where $\mathcal{L}_{prox}=\frac{\mu}{2}\|\theta-\theta_G^r\|_2^2$ controls client drift and $\mathcal{L}_{smooth}$ penalizes abrupt score changes when adjacent windows have consistent identities and traces.

### E. RISK-AWARE FEDERATED AGGREGATION

For each update, the coordinator calculates three bounded factors. Coverage $q_i$ reflects the diversity of locally observed security categories using a privacy-preserving histogram; consistency $c_i$ is the cosine agreement between the clipped local update and a robust median update; reliability $v_i$ is the improvement on a small shared validation set containing sanitized graph statistics rather than raw requests.

The aggregation weight is

$$\begin{aligned} r_i &= \beta^\top [\log(1+n_i), q_i, c_i, v_i], \\ \omega_i &= \frac{\exp(r_i)}{\sum_{j\in C_r}\exp(r_j)}, \\ \theta_G^{r+1} &= \theta_G^r + \eta \sum_{i\in C_r} \omega_i \,\mathrm{clip}(\Delta\theta_i^r). \end{aligned} \tag{10}$$

The logarithm prevents a high-volume client from dominating solely because of request count. Coverage increases the influence of rare but useful attack evidence. Consistency and reliability reduce the weight of unstable or poorly generalizing updates.

**TABLE I.** Controlled API Replay Corpus

| Split | Requests | Normal | Malicious | Sessions |
|---|---|---|---|---|
| Training | 868,000 | 808,920 | 59,080 | 31,640 |
| Validation | 186,000 | 173,460 | 12,540 | 6,780 |
| Test | 186,000 | 172,980 | 13,020 | 6,810 |
| Total | 1,240,000 | 1,155,360 | 84,640 | 45,230 |

### F. ADAPTIVE RISK SCORE AND EVIDENCE PATH

At inference, three normalized components are fused:

$$\begin{aligned} R_t &= \rho_1(1-\hat{p}_{normal}) + \rho_2 d_t + \rho_3 b_t, \\ d_t &= \mathrm{norm}(\|z_t - c_i\|_2), \\ b_t &= \mathrm{norm}(|s_t - \bar{s}_t^{EWMA}|). \end{aligned} \tag{11}$$

The alert threshold is $\tau_t=Q_{1-\delta}(R_{t-h:t})+\kappa\cdot\mathrm{MAD}(R_{t-h:t})$, where the rolling quantile and median absolute deviation adapt to local load without allowing a single burst to reset the baseline. Adjacent alerts sharing subject, endpoint path, or trace identifier are grouped into one incident. The explanation retains the top-attended nodes, relation edges, and time windows, providing an evidence path for analyst review.

## V. EXPERIMENTAL METHODOLOGY

### A. CONTROLLED REPLAY CORPUS

The evaluation used a controlled API replay environment rather than production interception. Twelve containerized services implemented identity, asset, file, configuration, query, notification, and audit functions. Benign traffic was generated from recorded workflow templates with randomized identities, object classes, and timing. Attack scripts implemented six families aligned with common API risks: object-level authorization abuse, authentication/token misuse, resource exhaustion, unrestricted business-flow abuse, server-side request forgery, and inventory/version probing.

The corpus contains 1,240,000 requests and 40 monitoring items. It was partitioned by session into 70% training, 15% validation, and 15% testing data to prevent adjacent windows from crossing splits. Twenty simulated participants received non-IID traffic using a Dirichlet distribution with concentration $\alpha$=0.5. No participant received all six attack families (Table I).

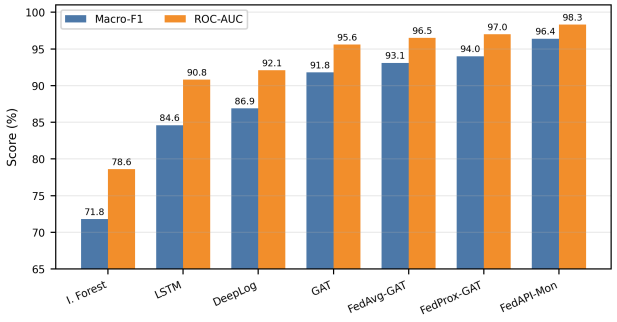


**Fig. 2.** Macro-F1 and ROC-AUC on the controlled API replay test set.

**TABLE II.** Main Detection Performance

| Method | Precision (%) | Recall (%) | Macro-F1 (%) | AUC (%) | FPR (%) |
|---|---|---|---|---|---|
| Isolation Forest | 70.1 | 73.6 | 71.8 | 78.6 | 9.8 |
| LSTM | 85.8 | 83.5 | 84.6 | 90.8 | 6.1 |
| DeepLog | 88.2 | 85.7 | 86.9 | 92.1 | 5.4 |
| GAT | 92.6 | 91.0 | 91.8 | 95.6 | 3.8 |
| FedAvg-GAT | 93.8 | 92.4 | 93.1 | 96.5 | 3.4 |
| FedProx-GAT | 94.7 | 93.4 | 94.0 | 97.0 | 3.0 |
| FedAPI-Mon | 96.9 | 95.9 | 96.4 | 98.3 | 1.7 |

### B. BASELINES AND CONFIGURATION

The baselines were Isolation Forest, an LSTM sequence classifier, DeepLog-style next-event detection [3], centralized GAT, FedAvg-GAT, and FedProx-GAT. All neural baselines used the same event features where their input structure allowed. FedAPI-Mon used two graph-attention layers with four heads, a two-layer temporal encoder, hidden dimension 128, sequence length 12 windows, and dropout 0.2. The federation contained 20 clients; 8 were sampled per round for 80 rounds. Local training used AdamW, learning rate $2\times10^{-4}$, batch size 64, and two local epochs.

Each reported value is the mean of five runs with different seeds. The primary metrics are precision, recall, macro-F1, ROC-AUC, and false-positive rate (FPR). Unknown-attack performance is evaluated by withholding one entire attack family from training and treating it as anomalous at test time. Alert compression is the percentage reduction after incident grouping, and malicious-event retention is the percentage of malicious source events retained in at least one grouped incident.

## VI. RESULTS AND ANALYSIS

### A. MAIN DETECTION PERFORMANCE

Table II and Figure 2 show that FedAPI-Mon achieves the best overall balance. Compared with centralized GAT, macro-F1 improves from 91.8% to 96.4%, indicating that the gain is not explained by graph representation alone. Compared with FedProx-GAT, the proposed method adds 2.4 percentage points of macro-F1 and reduces FPR from 3.0% to 1.7%. The difference is associated with temporal modeling, boundary learning, and security-aware update weighting.

The ROC curves in Figure 3 confirm that the improvement is not tied to one operating threshold. FedAPI-Mon maintains a higher true-positive rate in the low-FPR region,

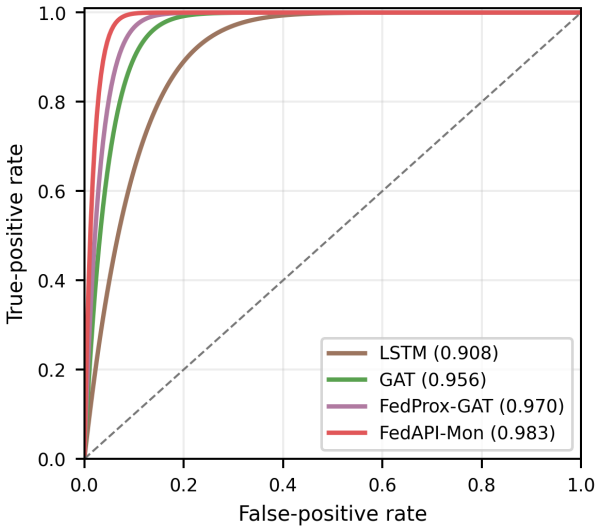


**Fig. 3.** Receiver-operating-characteristic curves for representative methods.

**TABLE III.** Leave-One-Attack-Family-Out Results

| Withheld family | FedAvg-GAT F1 | FedProx-GAT F1 | FedAPI-Mon F1 |
|---|---|---|---|
| Object authorization abuse | 84.9 | 87.3 | 92.6 |
| Authentication/token misuse | 83.7 | 86.5 | 91.4 |
| Resource exhaustion | 89.5 | 91.2 | 95.1 |
| Business-flow abuse | 80.8 | 83.6 | 89.2 |
| Server-side request forgery | 86.1 | 88.0 | 92.4 |
| Inventory/version probing | 85.2 | 87.4 | 90.3 |
| Average | 85.0 | 87.3 | 91.8 |

which is operationally important because normal API traffic is substantially more frequent than malicious traffic.

### B. UNKNOWN-ATTACK GENERALIZATION

In the leave-one-attack-family-out evaluation, all samples of one family were excluded from training and were introduced only during testing. Table III reports the binary anomaly-detection result for each withheld family. The average macro-F1 is 91.8%. Resource exhaustion is the easiest family because it changes timing and rate features. Business-flow abuse is the most difficult because individual calls are valid and the anomaly lies mainly in long-range order. The graph-temporal representation still retains 89.2% macro-F1 for this setting, outperforming FedProx-GAT by 5.6 points.

### C. ROBUSTNESS TO NON-IID TRAFFIC

Figure 4 varies the Dirichlet concentration parameter. Lower values create stronger client heterogeneity. FedAvg-GAT loses 7.2 macro-F1 points as $\alpha$ decreases from 10.0 to 0.1, while FedAPI-Mon loses 2.3 points. The risk-aware rule does not eliminate statistical heterogeneity, but it prevents volume-dominant, narrow-coverage updates from overwhelming rarer evidence.

### D. ABLATION STUDY

Figure 5 and Table IV isolate the contribution of each component. Removing the graph encoder produces the largest

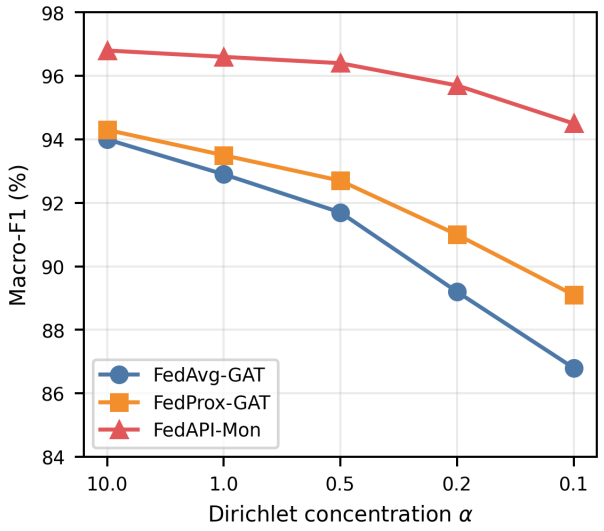


**Fig. 4.** Macro-F1 under different levels of client heterogeneity. Smaller Dirichlet concentration values indicate more non-IID traffic.

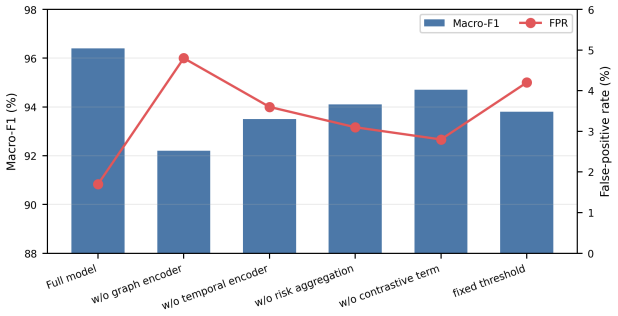


**Fig. 5.** Ablation results for representation, aggregation, boundary learning, and threshold adaptation.

**TABLE IV.** Ablation Results

| Configuration | Macro-F1 (%) | AUC (%) | FPR (%) |
|---|---|---|---|
| Full FedAPI-Mon | 96.4 | 98.3 | 1.7 |
| Without graph encoder | 92.2 | 95.4 | 4.8 |
| Without temporal encoder | 93.5 | 96.2 | 3.6 |
| Without risk-aware aggregation | 94.1 | 96.8 | 3.1 |
| Without contrastive boundary | 94.7 | 97.1 | 2.8 |
| Fixed global threshold | 93.8 | 97.6 | 4.2 |

macro-F1 decrease because endpoint and identity relations collapse into independent vectors. A fixed global threshold causes FPR to rise from 1.7% to 4.2%, showing that local baseline adaptation is essential even after federated model training. Removing risk-aware aggregation reduces macro-F1 by 2.3 points and makes the detector more sensitive to client composition.

### *E. ALERT COMPRESSION AND RUNTIME*

Grouping adjacent detections by subject, trace, endpoint path, and time interval reduces repeated alerts. Figure 6 shows the tradeoff as the grouping threshold changes. At the selected operating point, the method compresses 96.2% of raw alerts while retaining 95.4% of malicious source events.

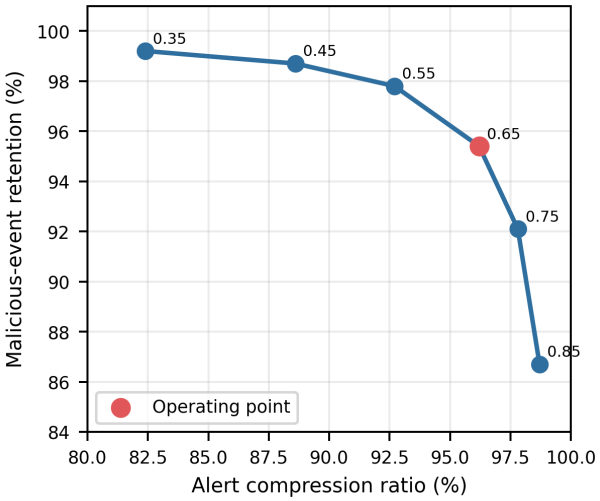


**Fig. 6.** Alert-compression and malicious-event-retention tradeoff. Labels indicate the grouping threshold.

**TABLE V.** Efficiency and Operational Indicators

| Indicator | Result |
|---|---|
| Training time, 80 rounds | 3.8 h |
| Inference throughput | 11,600 requests/s |
| P95 scoring latency | 83 ms |
| Communication/client/round | 6.8 MB |
| Alert compression ratio | 96.2% |
| Malicious-event retention | 95.4% |

This result should be interpreted as event consolidation rather than attack removal: each grouped incident maintains pointers to its original request windows.

On an RTX 3090 workstation, offline training required 3.8 hours for 80 rounds. Online inference processed 11,600 requests per second at batch size 256. End-to-end window scoring latency was 83 ms at the 95th percentile, including graph construction. Federated updates added 9.4% training time and 6.8 MB of communication per participating client per round. No raw request body or token was transmitted (Table V).

## VII. DISCUSSION

The results support three findings. First, API security behavior is both relational and temporal. Graph attention improves the representation of identity, object, and service dependencies, while temporal attention captures multi-step activity that is not anomalous at the individual-request level. Their combination is consistently stronger than either component alone.

Second, federated aggregation should reflect evidence quality. Sample count is a weak proxy for security value because a smaller domain may observe the only instance of a rare abuse pattern. The proposed coverage factor gives such evidence a controlled influence, while consistency and validation reliability prevent rare data from automatically receiving a high weight. The design does not require the

coordinator to inspect raw requests.

Third, a global learned representation and a local adaptive boundary are complementary. The global model transfers attack knowledge across participants, but local quantiles and median absolute deviation account for differences in request volume and business rhythm. This separation reduces false positives without fragmenting the feature encoder into unrelated local models.

Several limitations remain. The corpus was generated through controlled replay and does not establish field effectiveness in a production grid enterprise. Encrypted payload semantics were represented through local schema and metadata features rather than full content inspection. The current evaluation assumes honest-but-curious coordination and bounded update noise; Byzantine clients, backdoor poisoning, and formal differential privacy require additional study. Finally, attention weights provide a useful evidence path but should not be treated as a complete causal explanation. A production deployment should combine model evidence with authorization policy, asset inventory, vulnerability information, and analyst confirmation.

## VIII. CONCLUSION

This paper presented FedAPI-Mon, a risk-aware federated graph-temporal method for cross-domain API security monitoring. The method constructs privacy-preserving call graphs, learns service and identity relations with graph attention, models multi-step behaviors with temporal self-attention, and aggregates distributed updates according to threat coverage, consistency, and validation reliability. An adaptive boundary combines supervised confidence, prototype distance, and local baseline deviation to support known and previously withheld attacks. Controlled replay results show 96.4% macro-F1, 98.3% ROC-AUC, a 1.7% false-positive rate, and 91.8% macro-F1 under leave-one-attack-family-out evaluation. Future work will evaluate the method on long-duration production traces, add secure aggregation and poisoning resistance, and study policy-aware explanations for analyst review.

## ACKNOWLEDGMENT

This work was supported by the Yunnan Power Grid Co., Ltd. Science and Technology Project, “Research on Vulnerability Lifecycle Governance and Security Protection Technologies in Cloud Environments” (Project No. YNKJXM20240458), under the subproject “Research on Intelligent Cross-Domain Cloud Asset Inventory and Vulnerability Governance Technologies,” and by the Technical Service Project of Puer Power Supply Bureau, Yunnan Power Grid Co., Ltd., “Research on Distributed Information Asset Vulnerability Risk Governance Technologies” (Contract No. 05080020250303010500373).

## REFERENCES

[1] OWASP Foundation, “OWASP Top 10 API Security Risks – 2023,” OWASP API Security Project, 2023. [Online]. Available: https://owasp.org/API-Security/editions/2023/en/0x11-t10/.

[2] H. B. McMahan, E. Moore, D. Ramage, S. Hampson and B. Agüera y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in Proc. 20th Int. Conf. Artif. Intell. Stat., 2017, pp. 1273–1282.

[3] M. Du, F. Li, G. Zheng and V. Srikumar, “DeepLog: Anomaly detection and diagnosis from system logs through deep learning,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2017, pp. 1285–1298, DOI: 10.1145/3133956.3134015.

[4] P. Veličković et al., “Graph attention networks,” in Proc. 6th Int. Conf. Learn. Represent., 2018.

[5] A. Vaswani et al., “Attention is all you need,” in Proc. Adv. Neural Inf. Process. Syst. 30, 2017, pp. 5998–6008.

[6] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar and V. Smith, “Federated optimization in heterogeneous networks,” in Proc. Mach. Learn. Syst., vol. 2, 2020, pp. 429–450.

[7] S. P. Karimireddy et al., “SCAFFOLD: Stochastic controlled averaging for federated learning,” in Proc. 37th Int. Conf. Mach. Learn., 2020, pp. 5132–5143.

[8] T.-Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, “Focal loss for dense object detection,” in Proc. IEEE Int. Conf. Comput. Vis., 2017, pp. 2999–3007.

[9] W. L. Hamilton, R. Ying and J. Leskovec, “Inductive representation learning on large graphs,” in Proc. Adv. Neural Inf. Process. Syst. 30, 2017, pp. 1024–1034.

[10] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in Proc. 3rd Int. Conf. Learn. Represent., 2015.