Submitted to
International Conference on Emerging Trends in Intelligent Computing and Cyber Security (ICETCCS 2025)
ESP Journal of Engineering & Technology Advancements (ESP-JETA)
ISSN: 2583-2646 / Volume 5 Issue 2 May 2025
Paper Id: JETA-V5I2P137 / Page No: 342-351

Original Article

Request-Level Anomaly Detection and Risk-Adaptive Authorization for Microservice APIs

Rajesh Balaji
Independent Researcher, Seattle, WA, USA
Anna University, Chennai, India

Received Date: 30 May 2025 *Revised Date: 11 June 2025* *Accepted Date: 25 June 2025*

Abstract: *A request can satisfy an authorization policy and still be harmful. A support agent may be permitted to issue refunds but misuse that authority; a customer can scrape public catalogue data; and a stolen session can present a valid token. We investigate whether a microservice API can use the decisions and permissions already encoded in its authorization policy to recognize such behavior and apply a proportionate restriction. Our method aggregates gateway and Open Policy Agent (OPA) decision records into five-minute windows, combines six gateway rules with role-specific Isolation Forest scores, and scales anomaly evidence according to the sensitive actions available to each principal. Depending on risk and the type of signal, the proposed response is a step-up challenge, a restricted session, or session revocation.*

The evaluation uses a synthetic AegisCommerce retail-API workload with policy-derived allow/deny decisions. Across five held-out test seeds, approximately one million generated requests include 200 modeled incidents spanning eight scenarios. The combined detector achieves mean window-level F1 of 0.748 and incident recall of 0.885, compared with 0.672 and 0.785 for gateway rules and 0.323 and 0.535 for Isolation Forest alone. It produces 4.3 false alert episodes per day on average. In a separately simulated response model, graded actions stop 70.2% of modeled attack activity, compared with 74.5% for a 60-minute block on the same alerts, while reducing legitimate requests blocked from 49.9 to 33.8 per day and avoiding complete benign-user lockouts under the assumed challenge outcomes. These findings establish behavior within the specified simulation; production effectiveness and user impact remain unmeasured.

Keywords: *API Security, Anomaly Detection, Insider Threat, Risk-Based Authentication, Step-Up Authentication, Continuous Access Evaluation, Microservices, Open Policy Agent, Zero Trust*

I. INTRODUCTION

Authorization answers whether a request is permitted; it does not necessarily establish whether the request is benign. In the modeled retail API from our companion study [25], a support agent may legitimately issue refunds and inspect customer records. The same permissions could be used for repeated unauthorized business activity. A customer can repeatedly read catalogue pages without violating an object-level access rule, and a stolen session may satisfy ordinary token checks. Detecting these patterns requires evidence accumulated across requests rather than a decision about one request in isolation. This study therefore examines how policy decisions, behavior over time, and response costs interact in a controlled API-security evaluation.

In practice, API abuse detection sits at the gateway or in a web application firewall and is mostly rule-based. Rules count login failures per source, requests per minute and forbidden responses, and they act by blocking a client or a user for a fixed period. They are easy to reason about, and on a well-authorized API they work better than they used to. Under a strict policy, many attacks produce a burst of denied requests, and that burst is a clean signal. What rules cannot see is abuse that the policy allows. Their response is also blunt: a block that stops a credential-stuffing bot also locks out the customer whose phone happened to trip the threshold.

We ask two questions. First, can a behavioral detector improve recognition of allowed-but-abusive activity when the authorization policy supplies an estimate of what a principal can reach? Second, does choosing a response based on the signal, rather than blocking every alerted principal, reduce disruption to legitimate traffic? We test these questions on AegisCommerce, a modeled seven-service retail API. We use OPA-derived authorization outcomes, synthetic behavior and incident scenarios, and an explicit response simulator. The design is not presented as completed production gateway integration.

This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/2.0/)

The contributions are as follows.

- A request-level telemetry model for microservice APIs. Per-principal and per-client windows summaries request volume, object breadth, policy denials, authentication failures, session context and sensitive actions.
- An exposure-weighted risk score that combines gateway rules and per-role anomaly detection, with each principal's exposure computed from the authorization policy rather than assigned by hand.
- A graded, risk-adaptive authorization response (step-up, read-only, revoke) and a method for measuring its cost as legitimate requests blocked and challenges imposed on real users.
- A reproducible comparison of rule-based detection, an unsupervised ML detector and the risk-adaptive approach on precision, recall, false-positive rate, detection latency and enforcement cost. It uses synthetic API traffic whose allow and deny outcomes are decided by the real policy from [25].

The study uses generated request traces rather than production traffic. The source implementation and generator are necessary for independent replication; availability should be confirmed and a stable repository or archive identifier inserted before submission. Section VIII identifies the deployment and user-study evidence that remains outstanding.

Section II reviews related work. Section III sets out the problem and Section IV the pipeline. Section V describes the detectors, the risk score and the responses. Sections VI and VII cover the experiment and results, and Section VIII discusses limitations and concludes.

II. BACKGROUND AND RELATED WORK

A. API Abuse

The OWASP API Security Top 10 [2] separates abuse that authorization should stop, such as broken object-level authorization, from abuse it cannot stop by design. In the second group are unrestricted resource consumption and unrestricted access to sensitive business flows: buying out stock, mass account creation, scraping. OWASP's catalogue of automated threats [3] and its credential-stuffing guidance [16] describe the latter in detail, including credential stuffing, scraping and account aggregation. Thomas et al. [9] showed at scale how stolen credentials, from breaches, phishing and malware, are turned into account takeovers, which makes "a valid session used by the wrong person" one of the most common starting points for API abuse.

B. Risk-Based Authentication and Continuous Access Evaluation

Passwords alone are a weak gate [22], and most APIs rely on OAuth 2.0 tokens [6] issued after login. Risk-based authentication scores each login by its context, such as device, location and network, and asks for more proof when the score is high. Freeman et al. [7] gave a statistical model for it, and Waffling et al. [8] measured how it is used in practice. NIST SP 800-63B [4] sets requirements for throttling failed logins and for reauthentication. Two standards make in-session responses practical: the OAuth step-up authentication challenge [23], which lets an API tell a client that the current token is not strong enough for this request, and token revocation [24]. The OpenID Continuous Access Evaluation Profile [5] lets services share signals such as "session compromised" so that access is reconsidered during a session, not only at login, as Zero Trust architecture requires [1]. These provide the response mechanisms. They do not say when to use which.

C. Behavioral and Insider Detection

User and entity behavior analytics builds baselines of normal activity and flags departures. The insider-threat literature, surveyed by Salem et al. [14], and CERT's practical guidance [15] both stress that the most damaging insiders act within their permissions, which is exactly where authorization cannot help. Isolation Forest [10] is a common unsupervised baseline because it needs no labelled attacks. The well-known cautions still apply: anomaly is not malice [11], ML security evaluations often suffer from sampling bias and unrealistic base rates [12], and analysts drown in false positives [13]. Triage methods that rank alerts by context, such as No Doze [19], reduce that burden. NIST SP 800-30 [20] frames risk as likelihood times impact, which is the structure our score follows.

D. Position of This Work

Existing standards specify authorization, reauthentication, token revocation, and continuous access evaluation [1], [5], [23], [24]. Work on anomaly detection and alert prioritization likewise provides established techniques [10]–[13], [19]. The narrower question studied here is whether authorization-policy reachability can serve as an impact proxy when combining rule and anomaly evidence, and whether a signal-dependent response can lower simulated user disruption. We do not claim that

Isolation Forest, step-up authentication, or risk-based access control is new. The contribution is their policy-informed coupling and the comparative evaluation under a declared workload model.

III. PROBLEM SETTING

A. Application And Policy

AegisCommerce is the seven-service, 16-endpoint retail API reference model introduced in [25]. Its OPA policy distinguishes customers, support agents, and anonymous clients and defines ownership- and role-dependent permissions. In this paper, the same model provides the request vocabulary and authorization decisions. The evaluation does not demonstrate deployment of all seven services or integration of the detection pipeline into a live Kubernetes cluster.

B. Telemetry

The generated request records include principal or anonymous-client identifiers, session context, route, object-ownership relationship, authorization outcome, and indicators for device or network novelty. We aggregate these records into five-minute principal/client windows and compute the 13 features in Table I. These are features of the simulation schema. A real deployment would need to confirm that the gateway and OPA logging pipeline supplies them reliably, especially authentication failures, device changes, and object relationships. Personal identifiers should be pseudonymized and raw bearer tokens excluded.

Table 1: Window Features

Group	Features
Volume	Requests; distinct endpoints; catalogue reads
Breadth	Requests for objects owned by someone else; profile reads of other users
Policy outcome	Denied requests; image imports with a blocked URL
Authentication	Failed logins
Sensitive actions	Writes; refunds
Session context	New device; new network; night-time (00:00 to 06:00)

C. Abuse Scenarios

Eight scenarios cover the main API abuse patterns (Table II). Four produce policy denials or failed logins: A1, A3, A4 and A8. Four uses only allowed requests: A2, A5, A6 and A7. Two are insiders acting within their permissions, A6 and A7, the hardest case for any control.

Table 2: Abuse Scenarios

ID	Scenario	Principal	Duration (min)	What the policy sees
A1	Credential stuffing	Anonymous client	15	About 40 failed logins a minute
A2	Account takeover with a stolen session	Customer	20	New device and network; profile change; orders; all allowed
A3	BOLA enumeration of order IDs	Customer	10	About 20 requests a minute, 90% for others' objects, denied
A4	Probing support and internal endpoints	Customer	6	About 10 function-level requests a minute, denied
A5	Catalogue scraping with an account	Customer	60	About 120 catalogue reads a minute, allowed
A6	Refund abuse at night	Support agent	30	Refunds and payment lookups between 00:00 and 05:00, allowed
A7	Harvesting customer profiles	Support agent	30	About 15 profile reads a minute in working hours, allowed
A8	SSRF attempts from a hijacked support session	Support agent	5	Image imports with internal URLs, denied; new device

D. Benign Activity That Looks Suspicious

The traffic model includes the everyday events that trip API detectors. Customers sometimes mistype a password or an order number. About 8% of sessions come from a new device and 8% from a new network, independently, as when customers travel or change phones. Support agents work daytime shifts and occasionally issue refunds. The test weeks add three events never seen in training: a day of three times normal customer traffic, a two-hour spike in support refunds after a bad delivery

day, and a buggy mobile release that sends an unexpected field on profile updates and retries it, producing bursts of denials from 30 innocent customers.

E. Goals

A useful system should detect both denial-producing and allowed-but-abusive scenarios and keep false alerts to a handful a day. It should respond in a way that stops attack activity without locking out real users. We therefore measure detection quality and response cost separately.

IV. PIPELINE

The pipeline reads the decision logs the gateway and policy engine already write and returns one of three proportionate responses to the same gateway (Fig. 1). The authorization policy appears twice: once as the source of allow and deny decisions, and once as the source of each principal's exposure.

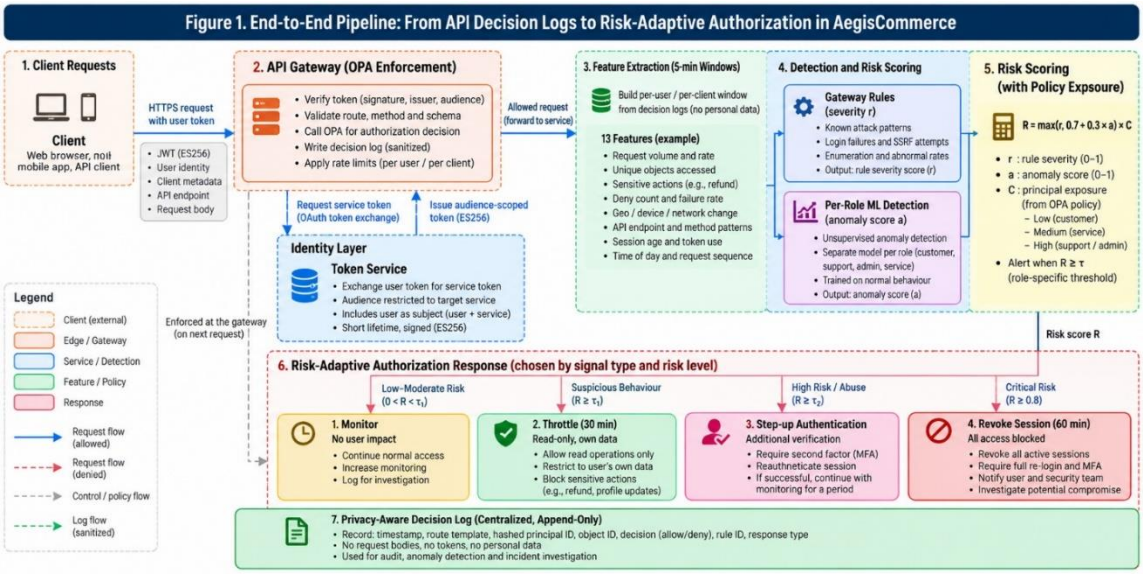


Figure 1: Proposed Architecture for Request-Level Anomaly Detection and Risk-Adaptive Authorization in Aegiscommerce.

API decision logs are aggregated into five-minute behavioral windows and evaluated using gateway rules and role-specific anomaly models. A policy-derived exposure factor adjusts the risk score, which determines whether a session is challenged, throttled, or revoked.

Every five minutes, each active user's or anonymous client's requests are summarized into a window of 13 features. Two detectors read the window. Gateway rules report the highest matching severity, and a per-role behavioral model reports how unusual the window is for that kind of principal. The risk score combines them and weights the anomaly part by the principal's exposure, which is computed once from the policy and recomputed whenever the policy changes. If the risk crosses the alert threshold, the response is chosen from the risk level and the kind of signal. At the top of the scale, the principal's sessions are revoked. Below that, a signal that the account may be in the wrong hands leads to a step-up challenge, and anything else, meaning abuse by someone who really is the account holder, leads to a throttle. The gateway enforces the response on the next request.

The architecture uses policy decisions in two roles: to generate allow/deny outcomes and to estimate each principal class's permitted sensitive actions. The experimental detector and response logic operate on generated traces. Actual enforcement of step-up challenges, session throttles, and revocations at a gateway is a proposed integration step, not a measured deployment result. Device and network context would also require an authenticated and privacy-reviewed collection mechanism in a real system.

V. DETECTORS, RISK SCORE AND RESPONSES

A. Gateway rules

The rule-based baseline uses six thresholds of the kind found in API gateways and web application firewalls (Table III). Each has a severity, which the risk score uses later. Any match raises an alert.

Table 3: Rules

Rule	Severity
Five or more failed logins from one client in a window	0.6
Five or more denied requests in a window	0.7
More than 300 requests in a window	0.5
Three or more refunds in a window	0.6
New device and new network in the same window	0.4
Any image import with a blocked URL	0.8

B. Behavioral detector

The ML baseline trains one Isolation Forest [10] per role (customer, support, anonymous), with 200 trees, on three days of attack-free windows, using scikit-learn [17]. Features are log-transformed. A window's anomaly is its percentile in the training score distribution of its role, and an alert is raised above a threshold q chosen on validation data. Modelling roles separately matters. A support agent reading 20 profiles an hour is normal; a customer doing the same is not.

C. Exposure from the authorization policy

The impact half of risk comes from the policy itself. We ask Open Policy Agent [18] which sensitive requests each kind of principal may make across the application's objects. These are reads of orders, payments and profiles, order status changes, refunds and support search. A support agent may make 24 of them, a customer 7 and an anonymous visitor none. The context weight is $C = 0.35 + 0.65 \times$ (exposure / maximum exposure), giving 1.00 for support agents, 0.54 for customers and 0.35 for anonymous clients. When the policy changes, the weights change with it.

D. Risk score

For each window, let r be the highest matching rule severity and a the anomaly tail score. The tail score is 0 at or below the 99th training percentile and rises linearly to 1 at the training maximum. It counts only if the same principal's previous window was also anomalous. The risk is:

$$R = \max\big(r \cot (0.7 + 0.3\,C),\; a \cot C\big)$$

The structure follows one principle. A rule match is strong evidence and keeps at least 70% of its severity whoever triggers it. An anomaly is weak evidence and is scaled fully by exposure, so the same unusual behavior counts for almost three times as much from a support agent as from an anonymous client. The threshold τ, and whether the two-window persistence is used, are chosen on validation data by window-level F1.

E. Graded responses

An alert leads to one of three responses, chosen by how high the risk is and what kind of signal produced it:

- **Revoke** ($R \geq 0.8$): end all of the principal's sessions and block it for 60 minutes. This is token revocation [24] combined with a continuous-access-evaluation signal [5].
- **Step-up** ($\tau \leq R < 0.8$, with an identity signal such as a new device and network, or failed logins): challenge the current session for a second factor [23], or a CAPTCHA for anonymous clients. Benign users pass; a stolen session or a bot does not.
- **Throttle** ($\tau \leq R < 0.8$, without an identity signal): for 30 minutes, make the session read-only, deny access to objects owned by others, and cap it at 10 requests a minute.

The split between step-up and throttle came from development data, and it reflects a real limit of step-up authentication. A challenge only helps when the problem is who is using the account. An insider issuing refunds, or a customer scraping with their own account, passes the challenge without effort. For those, the response has to limit what the session can do. The rule-based and ML baselines use the common blunt response instead: block the principal for 60 minutes on any alert.

VI. EXPERIMENTAL SETUP

A. Traffic

All reported traffic is generated, not collected from production. The model includes 400 customers, six support agents, and approximately 1,500 anonymous visitors per simulated day. Customers initiate an average of 0.6 sessions per day; support agents follow modeled shifts. A test week contains roughly 200,000 requests and 26,000 five-minute principal windows. The generator draws an authorization outcome from a decision table produced by OPA for the modeled request classes; it does not execute every generated request against a running application. Behavior, incidents, and response consequences are synthetic. This makes scenario labels reproducible but limits claims about operational accuracy.

The generator's parameters were fixed before any detector ran and were not changed afterwards. Synthetic traffic gives exact ground truth and full reproducibility. Its cost is that benign variety is limited to what we modelled, and that the same author designed the rules and the attacks, which favors the rules.

B. Splits And Protocol

Each run uses three independent datasets. Three attack-free days train the Isolation Forests. A validation week with three instances of each scenario chooses the ML threshold, the risk threshold and whether to use two-window persistence. A test week with five instances of each scenario, plus the three unseen benign events, is used only for reporting. We repeat this for five test seeds (5 to 9), giving 200 incidents and about a million requests, and report the mean and standard deviation. A stealth variant halves every attack's intensity while keeping the thresholds chosen at full intensity.

Development used a separate seed (0), never reported. Two changes were made on it. The first was the split between step-up and throttle responses described in Section V-E: step-up alone failed against insiders and own-account abuse. The second was adding the blocked-URL rule, which was in the original design but had been left out of the code. Neither change was made after seeing test results.

C. Metrics

- **Detection:** window-level precision, recall, F1 and false-positive rate; incident recall (incidents with at least one alert); detection latency (minutes from attack start to the end of the first alerted window); and false alert episodes per day.
- **Response:** attack activity stopped, the share of each incident's requests after the first response that were blocked, averaged over all incidents with undetected incidents counted as zero. Legitimate requests blocked per day. Benign users fully blocked per day. Step-up challenges and throttles imposed on benign users per day.

Step-up outcomes follow Table IV. Benign users pass 97% of challenges, and a failed challenge ends that session.

Table 4: Who Passes a Step-Up Challenge?

Attacker	Passes	Reason
Stolen session (A2), hijacked support session (A8)	Never	No access to the victim's second factor
Bot (A1)	10%	CAPTCHA-solving services
Own-account abuse (A3, A4, A5)	Always	The attacker owns the account
Insider (A6, A7)	Always	The attacker is the employee

D. Reproducing the Results

```
python3 decisions.py # decision table and exposure from OPA (needs .../app1)
python3 evaluate.py  # five test seeds, normal and stealth, about 2 minutes -> results. Son
```

VII. RESULTS

On the held-out test weeks, the exposure-weighted risk detector found 88.5% of incidents against 78.5% for the gateway rules and 53.5% for the ML detector, with the fewest false alerts of the three (4.3 episodes a day). Its graded responses stopped 70% of attack activity, close to the 75% a blunt block achieves on the same alerts. They blocked a third fewer legitimate requests than that blunt block and fully locked out no benign user. Rules lock out 5.4 a day.

A. Detection

The ML detector on its own is the weakest. It is unstable across seeds, producing between 2 and 23 false episodes a day depending on how the unseen events land. Most of its false alerts are anonymous visitors and customers doing something merely unusual. Combined with rules and weighted by exposure, the same anomaly signal raises incident recall by ten points over rules alone and cuts false episodes from 5.6 to 4.3 a day. The price is about one minute of extra detection latency.

Table 5: Detection on The Test Weeks (Mean ± SD Over Five Seeds; 200 Incidents)

Detector	Precision	Recall	F1	FPR	Incident recall	Latency (min)	False episodes / day
Gateway rules	0.667 ± 0.034	0.678 ± 0.014	0.672 ± 0.016	0.003	0.785 ± 0.020	3.6	5.6 ± 0.4
ML (Isolation Forest per role)	0.419 ± 0.153	0.300 ± 0.074	0.323 ± 0.027	0.004	0.535 ± 0.075	4.4	14.5 ± 7.5
Exposure-weighted risk	0.820 ± 0.010	0.690 ± 0.040	0.748 ± 0.023	0.001	0.885 ± 0.056	4.7	4.3 ± 0.4

B. Which Abuse Each Detector Sees

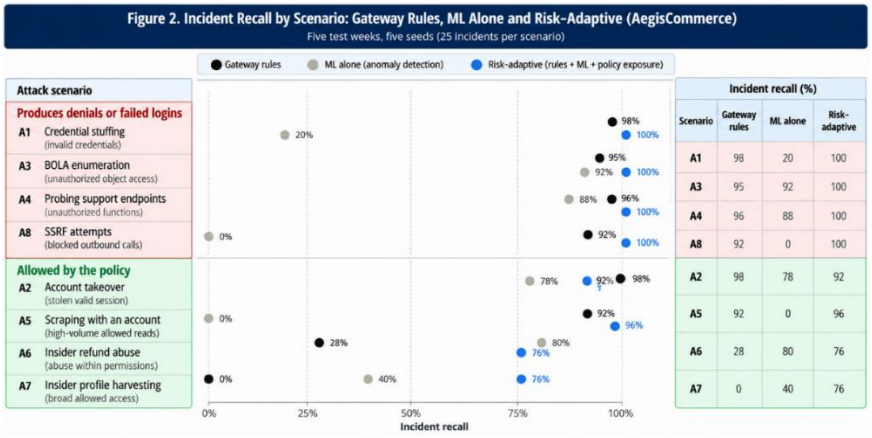


Figure 2: Breaks Incident Recall Down by Scenario.

Figure 2. Scenario-level incident recall for gateway rules, role-specific Isolation Forest detection, and exposure-weighted risk-adaptive detection across five synthetic test weeks. Each scenario contains 25 simulated incidents. Results illustrate the differing detection capabilities for policy-denied attacks and permitted-but-abusive API activity.

The pattern is clean. Every detector that includes the rules catches every scenario that produces denials or failed logins: credential stuffing, enumeration, probing and SSRF attempts. On a well-authorized API, the policy turns these attacks into loud signals. The difference lies in abuse the policy allows. Rules miss profile harvesting completely and catch night-time refund abuse only 28% of the time. The exposure-weighted detector catches 76% and 40% of these, because a support agent's anomalies count at full weight. The ML detector alone does better still on refunds (80%), but it misses scraping and SSRF entirely and pays for it with three times the false alerts. Profile harvesting during working hours remains the hardest case for every detector: at 15 reads a minute it looks like a busy shift. The weighting also costs something. Account takeover recall falls from 100% with rules to 92%, because a customer's new-device-and-network match is scaled down by the customer's lower exposure and sometimes falls just below the threshold.

C. Response Cost

Table 6: Effect of the responses (mean per day over the test weeks)

Detector and response	Attack activity stopped	Legitimate requests blocked / day	Benign users fully blocked / day	Benign step-ups / day	Benign throttles / day
Rules, 60-minute block	67.0%	74.7	5.4	–	–
ML, 60-minute block	47.5%	124.7	13.3	–	–
Risk, 60-minute block	74.5%	49.9	4.3	–	–
Risk, graded response	70.2%	33.8	0.0	0.26	4.2

The modeled workload contains approximately 22,800 legitimate requests per day. Under the response simulator, a 60-minute block applied to risk alerts stops 74.5% of modeled attack activity but blocks 49.9 legitimate requests and fully locks out 4.3 benign principals per day. Graded responses stop 70.2%, block 33.8 legitimate requests, and produce no complete benign-principal lockouts under the specified step-up success assumptions. These are simulated outcomes, not observed experiences of real users. A throttle still restricts legitimate actions and should not be treated as cost-free.

D. Ablation

Table 7: Removing the Exposure Weighting or The Graded Response

Variant	F1	Incident recall	False episodes / day	Attack activity stopped	Legitimate requests blocked / day	Benign users fully blocked / day
Full design	0.748	0.885	4.3	70.2%	33.8	0.0
Without exposure weighting (C = 1)	0.743	0.890	5.2	70.2%	37.0	0.6
Without graded response (60-minute block)	0.748	0.885	4.3	74.5%	49.9	4.3

The exposure weighting is a modest refinement. It leaves incident recall essentially unchanged but trims false episodes by about 17%. The false alerts it removes are almost all anonymous-client anomalies, which no longer clear the threshold once their low exposure is taken into account. The graded response is the larger effect on users, and it is a trade the operator chooses: a few points of attack activity for far fewer lockouts.

E. Stealthier Attacks

Table 8: Half-Intensity Attacks, Thresholds Unchanged

Detector	F1	Incident recall	Attack activity stopped
Gateway rules	0.542	0.750	65.0%
ML	0.325	0.570	53.0%
Risk, graded response	0.606	0.810	63.2%
Risk without exposure weighting	0.596	0.810	62.7%

All detectors lose window-level recall when attacks slow down, because a slower attack spreads over windows that individually look normal. The risk detector keeps the lead in incident recall, and the ordering of the three designs is unchanged.

VIII. DISCUSSION

A. What the Results Suggest

Three findings are worth carrying beyond this experiment. First, strict authorization and detection reinforce each other. With the policy from [25] in place, every attack that tries something forbidden produces denials, and simple rules catch all of them. This is a strong argument for fixing authorization before investing in detection. Second, what remains is abuse within permissions, and much of it comes from insiders and people using their own accounts. Weighting behavioral anomalies by what a principal can reach, a figure the policy already holds, is a cheap way to focus on the principals whose misuse would hurt most. Third, the response matters as much as the detection. A step-up challenge only answers "is this really you?", which is useless against someone who really is the account holder. Matching the response to the kind of signal let the same detector stop nearly as much attack activity as a blanket block while locking out no innocent users.

B. Limitations

- **Synthetic traffic.** The most important limitation. Benign behavior follows a model with a handful of parameters, and real API traffic is messier: retries, background syncs, partner integrations and seasonal swings. The comparison between designs is more trustworthy than any absolute figure.
- **Same author for rules and attacks.** This favors the rule-based detector, whose real-world recall against unfamiliar attackers is likely lower.
- **Simple behavioral model.** A per-role Isolation Forest over five-minute windows cannot see slow patterns, such as one refund an hour for a week. Per-principal baselines and longer horizons would help with exactly the insider cases where all detectors struggled.

- **Modelled step-up outcomes.** Pass rates in Table IV are assumptions, not measurements, and the friction a challenge causes users [21] is counted but not weighed. In particular, real MFA fatigue and phishing would let some stolen sessions pass.
- **Three roles.** Exposure weighting has only three distinct values here. In applications with fine-grained roles, or relationship-based permissions such as "agent assigned to this account", it would rank principals more finely.

C. Validating On Production Logs

The next validation stage should use appropriately authorized, privacy-reviewed operational logs and documented incident labels. Gateway and policy-decision records alone may be insufficient for the proposed features; device context, authentication outcomes, and object relationships must be checked. A shadow-mode deployment should first report incident recall, false-alert episodes, alert latency, and the number of users who would be challenged or restricted. Only after safety review should controlled enforcement trials measure actual challenge completion, request failures, service latency, and rollback behavior. None of those live outcomes is measured in this manuscript.

D. Next Steps

An alert with a graded response still needs someone to look at it: to confirm that a throttled support agent really was harvesting data, decide whether to revoke, and undo the throttle if not. That investigation is slow and repetitive, which makes it a natural target for AI agents. It is also where a manipulated agent could do real damage. Our follow-on work examines how to let agents investigate and respond to API incidents safely.

IX. CONCLUSION

This study examined a gap left by request-level authorization: misuse that remains within an account's permitted actions. On five held-out synthetic AegisCommerce workloads, combining gateway rules and role-specific anomaly detection with policy-derived exposure achieved mean incident recall of 0.885, compared with 0.785 for rules alone. The response simulation suggests that signal-dependent restrictions can reduce modeled benign-user disruption, although they stop somewhat less attack activity than a blanket block. The results depend on the selected scenarios, traffic assumptions, and modeled challenge success rates. Reproduction of the code and independent validation on operational data are needed before drawing conclusions about production effectiveness.

X. REFERENCES

[1] S. Rose, O. Burchett, S. Mitchell, and S. Connelly, "Zero Trust Architecture," NIST Special Publication 800-207, Aug. 2020.
[2] OWASP Foundation, "OWASP API Security Top 10 – 2023," owasp.org.
[3] OWASP Foundation, "OWASP Automated Threats to Web Applications," handbook, owasp.org.
[4] P. A. Grassy et al., "Digital Identity Guidelines: Authentication and Lifecycle Management," NIST Special Publication 800-63B, Jun. 2017 (updated Mar. 2020).
[5] Open ID Foundation, "Open ID Continuous Access Evaluation Profile 1.0," Shared Signals Working Group.
[6] Hard, Ed., "The Oath 2.0 Authorization Framework," RFC 6749, IETF, Oct. 2012.
[7] M. Freeman, S. Jain, M. Demuth, B. Baggie, and G. Jacinto, "Who Are You? A Statistical Approach to Measuring User Authenticity," in Proc. Network and Distributed System Security Symposium (NDSS), 2016.
[8] S. Waffling, L. Lo Dacono, and M. Demuth, "Is This Really You? An Empirical Study on Risk-Based Authentication Applied in the Wild," in Proc. IFIP International Conference on ICT Systems Security and Privacy Protection (SEC), 2019.
[9] K. Thomas et al., "Data Breaches, Phishing, or Malware? Understanding the Risks of Stolen Credentials," in Proc. ACM SIGSAC Conference on Computer and Communications Security (CCS), 2017.
[10] T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in Proc. 8th IEEE International Conference on Data Mining (ICDM), 2008.
[11] R. Summer and V. Parson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," in Proc. IEEE Symposium on Security and Privacy, 2010.
[12] Arp et al., "Dos and Don'ts of Machine Learning in Computer Security," in Proc. 31st USENIX Security Symposium, 2022.
[13] B. A. Alameda, L. Axon, and I. Martinovic, "99% False Positives: A Qualitative Study of SOC Analysts' Perspectives on Security Alarms," in Proc. 31st USENIX Security Symposium, 2022.
[14] M. B. Salem, S. Hershkop, and S. J. Stolfo, "A Survey of Insider Attack Detection Research," in Insider Attack and Cyber Security, Springer, 2008.
[15] Software Engineering Institute, "Common Sense Guide to Mitigating Insider Threats," 6th ed., Carnegie Mellon University, 2019.
[16] OWASP Foundation, "Credential Stuffing Prevention Cheat Sheet," cheatsheetseries.owasp.org.
[17] Pedregosa et al., "Sickie-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.
[18] Open Policy Agent Project, openpolicyagent.org.

[19] W. U. Hassan et al., "No Doze: Combatting Threat Alert Fatigue with Automated Provenance Triage," in Proc. Network and Distributed System Security Symposium (NDSS), 2019.
[20] Joint Task Force Transformation Initiative, "Guide for Conducting Risk Assessments," NIST Special Publication 800-30 Rev. 1, Sep. 2012.
[21] S. Waffling, M. Demuth, and L. Lo Dacono, "More Than Just Good Passwords? A Study on Usability and Security Perceptions of Risk-based Authentication," in Proc. Annual Computer Security Applications Conference (ACSAC), 2020.
[22] J. Bonnie, C. Harley, P. C. van Borscht, and F. Stefano, "The Quest to Replace Passwords: A Framework for Comparative Evaluation of Web Authentication Schemes," in Proc. IEEE Symposium on Security and Privacy, 2012.
[23] V. Bertocci and B. Campbell, "Oath 2.0 Step Up Authentication Challenge Protocol," RFC 9470, IETF, Sep. 2023.
[24] T. Norderstedt, Ed., S. Drone, and M. Scurtescu, "Oath 2.0 Token Revocation," RFC 7009, IETF, Aug. 2013.
[25] R. Bali, "Threat-Model-Driven Zero-Trust Authorization for Micro service APIs in Cabernets-Based Cloud-Native Applications," manuscript in preparation.