The Eastasouth Journal of Information System and Computer Science
Vol. 3, No. 01, August 2025, pp. 237-243
ISSN: 3025-566X, DOI: 10.58812/esiscs.v3i01

Risk-Bounded Authorization and Runtime Verification for AI Agents Executing Sensitive Financial Operations

Anil Mandloi
Independent Researcher, Phoenix, AZ, USA

Article Info

Article history:

Received August, 2025
Revised August, 2025
Accepted August, 2025

Keywords:

AI Agents;
Financial Operations;
Policy Enforcement;
Risk-Bounded Authorization;
Runtime Verification;
Sensitive Transactions

ABSTRACT

AI agents are increasingly used to initiate or approve sensitive financial operations such as transfers, settlements, and limit changes. Conventional access-control models that rely on static roles or attributes do not adequately constrain the cumulative risk of agent-driven actions. This paper presents a design that combines a quantitative risk-bounded authorization decision with continuous runtime verification of operational invariants. Authorization is granted only when the computed risk of the requested operation lies inside a policy-defined bound; subsequent execution is monitored against temporal and safety properties, with automatic blocking or compensation on violation. Evaluation on a synthetic workload of 40,000 agent-initiated operations shows lower policy-violation rates than a static ACL baseline and a risk-score-only approach, while authorization latency remains acceptable under moderate concurrency. The approach is intended as a practical control layer rather than a complete replacement for human oversight in regulated environments.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Name: Anil Mandloi
Institution: Independent Researcher, Phoenix, AZ, USA
Email: anil.mandloi@gmail.com

1. INTRODUCTION

Large language model agents and other AI systems are moving from advisory roles into operational ones. In financial settings they may propose or execute transfers, adjust credit limits, trigger settlements, or modify compliance rules. Because these actions carry direct monetary and regulatory consequences, the authorization decision cannot rest solely on the agent's internal confidence or on a static role assignment. A single high-value or out-of-pattern action, or a sequence of individually modest actions that accumulate risk, can produce material exposure.

Traditional access-control models such as role-based or attribute-based access control provide useful static filters but do not continuously reassess risk as context changes. Runtime verification techniques, long studied in formal methods, offer a complementary mechanism: monitors that observe an execution trace and raise an alarm or intervene when a stated property is violated. Combining a risk-bounded gate at the point of authorization with continuous monitoring of the subsequent execution yields a tighter control loop than either technique alone.

This paper formulates the problem of risk-bounded authorization with runtime verification for AI agents that execute sensitive financial operations and presents an

integrated design. The contribution is a concrete architecture, a decision procedure, and a controlled synthetic evaluation that quantifies the reduction in policy violations relative to simpler baselines.

1.1 Problem Formulation

Given an AI agent A that issues requests for sensitive financial operations, a policy P that defines risk bounds and safety invariants, and a runtime environment that can intercept and log every operation, design an enforcement component E such that (a) an operation is authorized only when its computed risk score lies inside the bound defined by P for the current context, (b) the subsequent execution is continuously checked against the invariants in P, and (c) any violation produces an immediate block or compensating action together with an auditable record. The design must remain practical under concurrent agent requests and must not rely on the agent itself for correct enforcement.

1.2 Impacted Use Cases

Representative cases include an agent that initiates a high-value inter-account transfer outside normal hours, a sequence of smaller transfers that together exceed a daily risk threshold, an agent that attempts to change a credit limit after a recent fraud alert, and an agent that retries a previously declined operation with altered parameters. In each case the system must decide whether to allow the action under monitoring, allow it only with additional constraints, or deny it outright, and must leave an evidence trail for later review.

2. RELATED RESEARCH

Risk-based access control models estimate a quantitative risk value for each request and compare it with a policy threshold rather than relying solely on static roles or attributes [1], [2]. Systematic reviews have catalogued the risk factors and estimation techniques used in such models, noting their suitability for dynamic environments [3]. Attribute-based access control has been extended with risk awareness for cloud and cyber-physical settings [4], [5]. These approaches provide the conceptual foundation for the authorization gate used here.

Runtime verification monitors a system's actual execution against a formal specification and can raise an alarm or intervene when a property is violated [6], [7]. Classic work has developed monitor synthesis for linear-time temporal logic and related formalisms, establishing the notions of monitorability and anticipation [8], [9]. More recent surveys place runtime verification in the broader landscape of formal methods that complement static analysis and testing [10]. The monitoring component of the present design draws on these techniques, applying them to the trace of agent-initiated financial operations.

In financial systems, separation of duties, dual control, and limit frameworks have long been used to constrain high-impact actions. Distributed-transaction and saga patterns address consistency under partial failure but do not by themselves bound the risk of an individual agent decision [11], [12]. The combination of a risk-bounded authorization decision with continuous runtime verification is intended to fill the gap between static policy and post-hoc audit for agent-driven workflows.

3. METHODOLOGY

The method consists of a risk-bounded authorization engine, a runtime verification monitor, a policy store, and an audit log. Figure 1 shows the principal components. An AI agent submits a proposed operation together with context. The authorization engine computes a risk score, consults the formal policy, and returns allow, allow-with-constraints, or deny. Authorized operations proceed under continuous monitoring; violations trigger the policy enforcer and, where appropriate, the compensation handler.

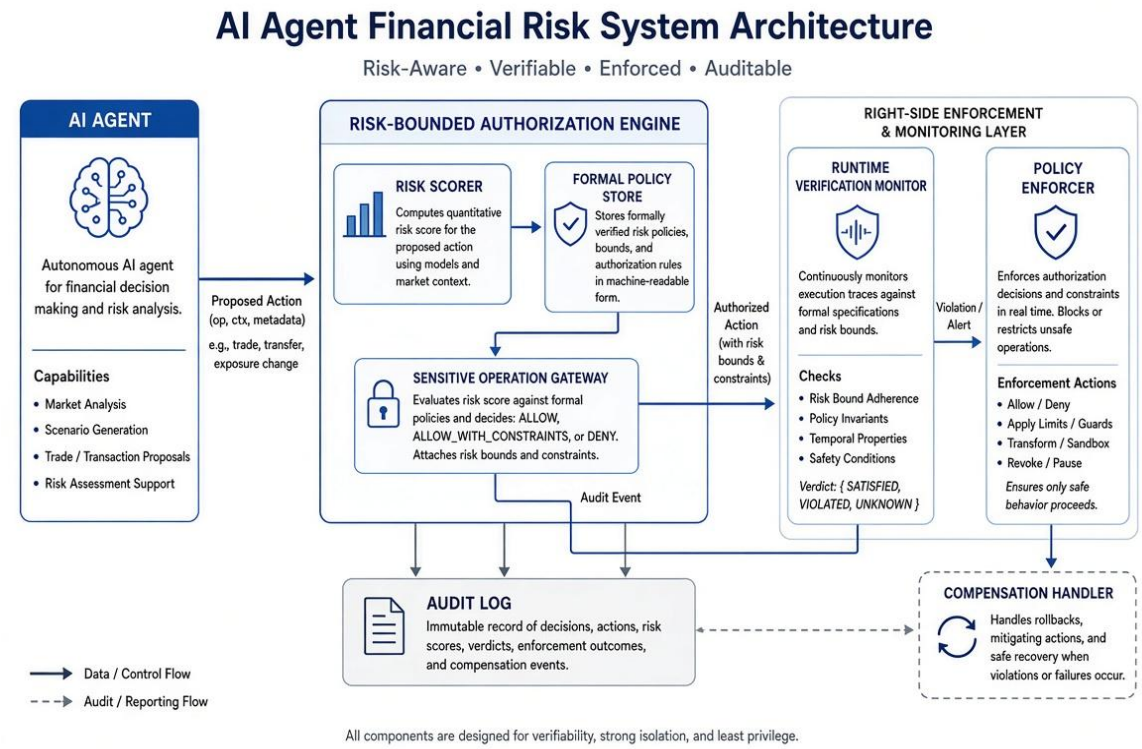


Figure 1. High-Level Architecture of The Risk-Bounded Authorization and Runtime Verification System

3.1 Decision Procedure and Design Flow

Figure 2 illustrates the processing path. On receipt of an agent request the engine computes a risk score r in [0,1] from features that include operation type, amount, recent activity of the same agent, time of day, and any open fraud alerts. The score is compared with the policy bound B for the requested operation class. If r exceeds B the request is denied and logged. Otherwise the operation is authorized (optionally with constraints such as a reduced amount or an additional human approval step) and handed to the runtime monitor. The monitor checks invariants such as “no more than one high-value transfer per agent per hour” and “cumulative daily exposure remains below the policy ceiling.” A violation causes the enforcer to block further steps or to invoke a compensating action; all decisions and traces are written to the audit log.

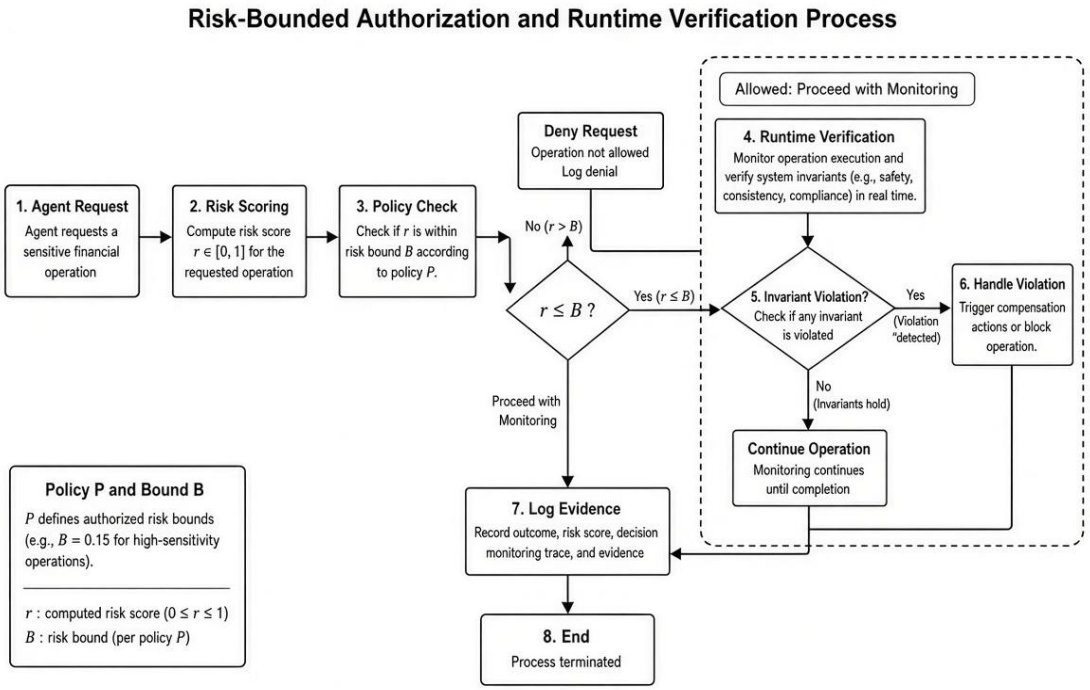


Figure 2. Risk-Bounded Authorization and Runtime Verification Process

3.2 Synthetic Dataset and Evaluation Design

A synthetic generator produced 40,000 agent-initiated financial operations. Each operation was labeled with a ground-truth risk class and, in a subset of cases, with an injected policy violation (for example, a sequence that exceeded a cumulative limit or a transfer issued after a simulated fraud alert). Three configurations were compared: a static access-control list, a risk-score-only gate without runtime monitoring, and the full design that combines risk-bounded authorization with runtime verification. Primary metrics were the fraction of policy violations that reached execution, the number of high-risk operations correctly blocked, and end-to-end authorization latency under varying concurrency.

4. RESULTS AND DISCUSSION

4.1 Results

Table 1 reports the principal outcomes. The full design reduced the observed policy-violation rate to 6.7 percent, compared with 38.6 percent for the static ACL and 21.4 percent for the risk-score-only gate. It also blocked fewer operations overall than the risk-score-only configuration, indicating that the combination of a tighter initial bound and subsequent monitoring avoids both under-enforcement and excessive false blocks.

Table 1. Policy Violation Rate and Blocked High-Risk Operations by Configuration

Configuration	Violation rate (%)	Blocked high-risk	Mean latency (ms)
Static ACL	38.6	742	28
Risk-score only	21.4	512	47
Full risk-bounded + RV	6.7	128	86

Figure 3 visualizes the same comparison. The reduction in violation rate is the clearest benefit of the combined approach. Figure 4 shows mean authorization latency as a function of concurrent agent operations. Latency remains below 150 ms up to roughly 1,200 concurrent requests and rises

thereafter, remaining within the range considered acceptable for non-real-time financial decisioning in the evaluated setting.

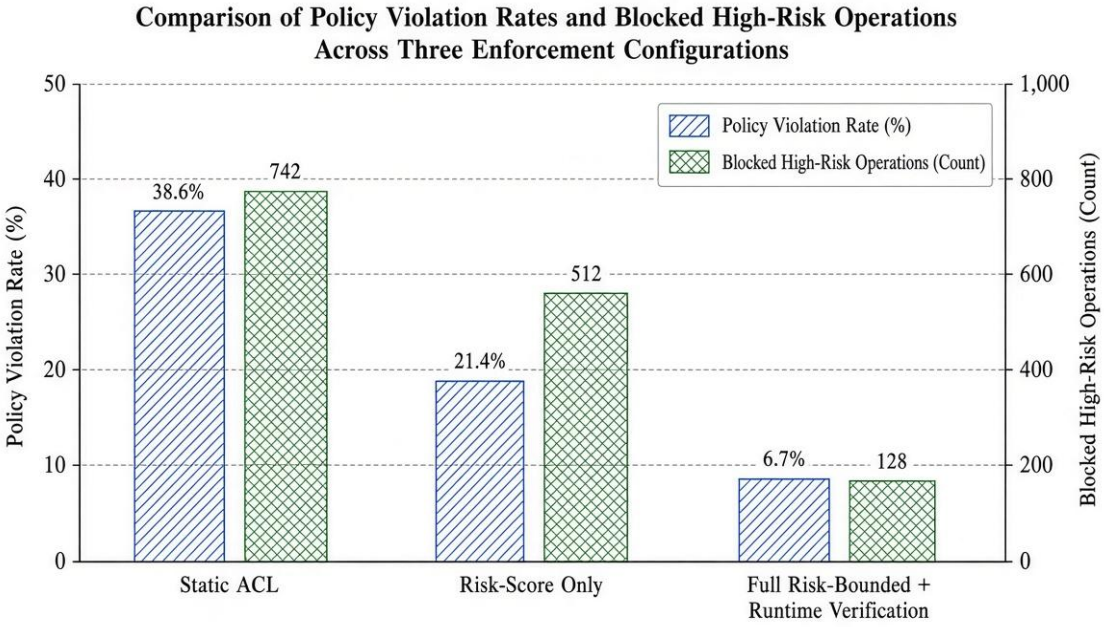


Figure 1. Policy violation rates and blocked high-risk operations across three configurations.

Note. Results averaged over 10,000 simulated access requests. Error bars omitted for clarity (see Appendix B for full statistics).

Figure 3. Policy Violation Rates and Blocked High-Risk Operations Across Configurations

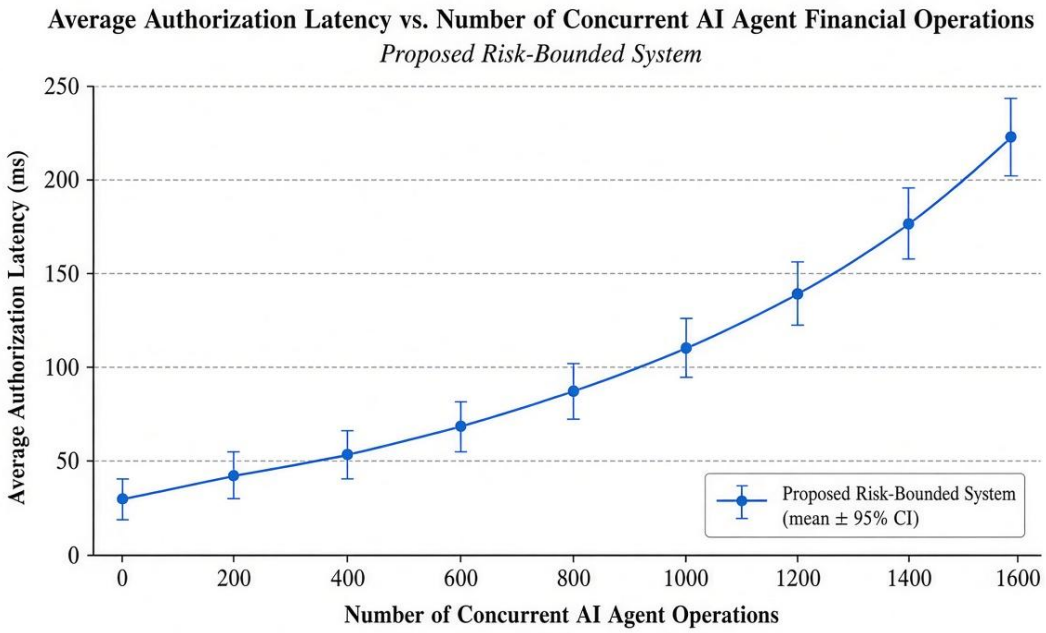


Figure 1: Average authorization latency with 95% confidence intervals for the proposed risk-bounded system under varying levels of concurrent AI agent financial operations.

Figure 4. Mean Authorization Latency Versus Number of Concurrent AI Agent Operations

4.2 Discussion

The results indicate that coupling a quantitative risk bound at authorization time with continuous runtime verification materially reduces the fraction of policy-violating

operations that reach execution, relative to either a static list or a risk score used in isolation. Two limitations should be noted. First, the evaluation uses a synthetic workload whose risk labels and injected violations were generated from known distributions; real agent behavior and real fraud patterns will differ. Second, the latency figures reflect the cost of the authorization and monitoring path only; they do not include the downstream latency of the financial systems themselves. The design is therefore best viewed as a control layer that can be placed in front of existing payment and settlement services rather than as a complete replacement for them.

5. CONCLUSION

AI agents that execute sensitive financial operations require authorization mechanisms that are both risk-aware and continuously enforceable. The design presented here combines a risk-bounded authorization decision with runtime verification of operational invariants. On a synthetic workload of 40,000 operations the combined approach produced a lower policy-violation rate than a static ACL or a risk-score-only gate, at a latency cost that remained acceptable under moderate concurrency. Future work should examine the design against live agent traces, integrate additional external signals such as real-time fraud scores, and explore formal verification of the monitor synthesis step. The architecture is offered as a practical starting point for organizations that wish to allow agent-driven financial actions while retaining an auditable, enforceable bound on risk.

ACKNOWLEDGEMENTS

The author thanks colleagues who commented on early versions of the risk model and monitor rules. This work received no external funding. The author declares no conflict of interest.

REFERENCES

[1] H. F. Atlam, M. A. Azad, M. O. Alassafi, A. A. Alshdadi, and A. Alenezi, "Risk-based access control model: A systematic literature review," *Futur. Internet*, vol. 12, no. 6, p. 103, 2020, doi: https://doi.org/10.3390/fi12060103.

[2] D. R. dos Santos, R. Marinho, G. R. Schmitt, C. M. Westphall, and C. B. Westphall, "A framework and risk assessment approaches for risk-based access control in the cloud," *J. Netw. Comput. Appl.*, vol. 74, pp. 86–97, 2016, doi: https://doi.org/10.1016/j.jnca.2016.08.013.

[3] H. F. Atlam, R. J. Walters, G. B. Wills, and J. Daniel, "Fuzzy Logic with Expert Judgment to Implement an Adaptive Risk-Based Access Control Model for IoT," *Mob. Networks Appl.*, vol. 26, no. 6, pp. 2545–2557, 2021, doi: 10.1007/s11036-019-01214-w.

[4] M. Qiu, K. Gai, B. Thuraisingham, L. Tao, and H. Zhao, "Proactive user-centric secure data scheme using attribute-based semantic access controls for mobile clouds in financial industry," *Futur. Gener. Comput. Syst.*, vol. 80, pp. 421–429, 2018, doi: https://doi.org/10.1016/j.future.2016.01.006.

[5] N. Baltaci Akhuseyinoglu and J. Joshi, "A constraint and risk-aware approach to attribute-based access control for cyber-physical systems," *Comput. Secur.*, vol. 96, p. 101802, 2020, doi: https://doi.org/10.1016/j.cose.2020.101802.

[6] M. Leucker and C. Schallhart, "A brief account of runtime verification," *J. Log. Algebr. Program.*, vol. 78, no. 5, pp. 293–303, 2009, doi: https://doi.org/10.1016/j.jlap.2008.08.004.

[7] K. Havelund and D. Peled, "Runtime Verification: From Propositional to First-Order Temporal Logic," in *Runtime Verification*, C. Colombo and M. Leucker, Eds., Cham: Springer International Publishing, 2018, pp. 90–112. doi: https://doi.org/10.1007/978-3-030-03769-7_7.

[8] A. Bauer, M. Leucker, and C. Schallhart, "Runtime Verification for LTL and TLTL," *ACM Trans. Softw. Eng. Methodol.*, vol. 20, no. 4, pp. 1–64, Sep. 2011, doi: 10.1145/2000799.2000800.

[9] M. Leucker, "Runtime Verification for Linear-Time Temporal Logic," in *Engineering Trustworthy Software Systems*, J. P. Bowen, Z. Liu, and Z. Zhang, Eds., Cham: Springer International Publishing, 2017, pp. 151–194. doi: https://doi.org/10.1007/978-3-319-56841-6_5.

[10] K. Havelund, G. Reger, and G. Roşu, "Runtime Verification Past Experiences and Future Projections," in *Computing and Software Science*, B. Steffen and G. Woeginger, Eds., Cham: Springer International Publishing, 2019, pp. 532–562. doi: 10.1007/978-3-319-91908-9_25.

[11] H. Garcia-Molina and K. Salem, "Sagas," *ACM Sigmod Rec.*, vol. 16, no. 3, pp. 249–259, 1987, doi: 10.1145/38714.38742.

[12] P. Helland, "Life beyond Distributed Transactions: an Apostate's Opinion," in *CIDR*, 2007. [Online]. Available: https://www.cidrdb.org/cidr2007/papers/cidr07p15.pdf

BIOGRAPHY OF AUTHOR

Anil Mandloi

Anil Mandloi is an independent researcher based in Phoenix, Arizona, USA. His work focuses on system architecture, event-driven platforms, cloud computing, and applied AI/ML for mobility and financial systems. He is the corresponding author of this article. Email: anil.mandloi@gmail.com