Business Risk Analytics for API-Driven Digital Platforms: Measuring Mass Assignment Exposure in Developer Workflows

Ananya Nair[1], Vivek Menon[2], Rajeev Thomas[3,*]
[1]Department of Business Analytics, University of Mysore, Mysuru 570006, Karnataka, India
[2]Department of Computer Applications, University of Calicut, Malappuram 673635, Kerala, India
[3]Department of Management Studies, Cochin University of Science and Technology, Kochi 682022, Kerala, India
*Email: rajeev.thomas@cusat.ac.in (Corresponding Author)

Abstract

API-driven digital platforms transform business processes into programmable services, but they also expose sensitive workflow states through request-response interfaces. Mass assignment exposure occurs when a server binds user-supplied fields directly to internal model objects without adequate field-level control, allowing unauthorized changes to roles, balances, ownership, verification states, or workflow status. This study develops a business risk analytics framework for measuring such exposure in developer workflows. The proposed Mass Assignment Exposure Index combines static indicators such as direct entity binding, sensitive-field setters, validation gaps, deserializer tolerance, cascade propagation, and service-layer copy logic with dynamic evidence from schema-aware injection tests. The Workflow Business Risk Score then weights exposure by asset value, identity sensitivity, regulatory impact, and remediation friction. A numerical evaluation based on three benchmark project classes and eight high-risk endpoint candidates shows that hybrid analytics reduces immediate actionability from eight static candidates to three confirmed high-impact exposures while preserving a backlog for fragile designs. Scenario analysis indicates that a hybrid risk policy prevents confirmed exposure without the workflow disruption produced by a static-only gate. The findings contribute to business and data analytics by converting API security telemetry into decision-oriented risk measures for release governance, internal audit, and platform control design.

Keywords: api-driven digital platforms; business risk analytics; mass assignment exposure; developer workflows; RESTful API security; static analysis; dynamic verification; risk scoring

Article History:

Received: April 18, 2023
Revised: June 08, 2023
Accepted: August 16, 2023
Available Online: September 30, 2023

See: https://inatgi.in/index.php/jbda/index for more information. https://doi.org/10.63646/jbda.2023.010303

Business Risk Analytics for API-Driven Digital Platforms: Measuring Mass Assignment Exposure in Developer Workflows

1. Introduction

API-driven digital platforms rely on continuous interaction among users, developers, partner systems, payment services, identity providers, and data stores. This interaction is productive because it allows a platform to release new functions quickly, expose programmable services to partners, and turn internal data resources into reusable business capabilities. The same interaction also produces a fragile control environment. When application code binds an incoming request body directly to an internal model object, a client may submit fields that the server did not intend to accept. In a business workflow, the issue is not merely a technical defect. It becomes an exposure channel through which price privileges, user roles, account balances, approval flags, partner identifiers, or order states move outside the intended authorization boundary.

Mass assignment exposure is especially difficult to govern because vulnerable code often resembles ordinary code used by productive development teams. A controller that accepts an object, a model class with generated setters, a service method that copies request fields into a persistence object, and a permissive deserialization configuration all appear normal in a fast-moving application team. The uploaded source paper describes this structural challenge in Spring Boot workflows: static analysis detects risky patterns, while dynamic verification confirms which endpoints actually accept injected fields in a running application. Its empirical result is also useful for business analytics: three out of eight high-risk endpoint candidates were dynamically confirmed, meaning that static alerts require economic triage rather than mechanical escalation.

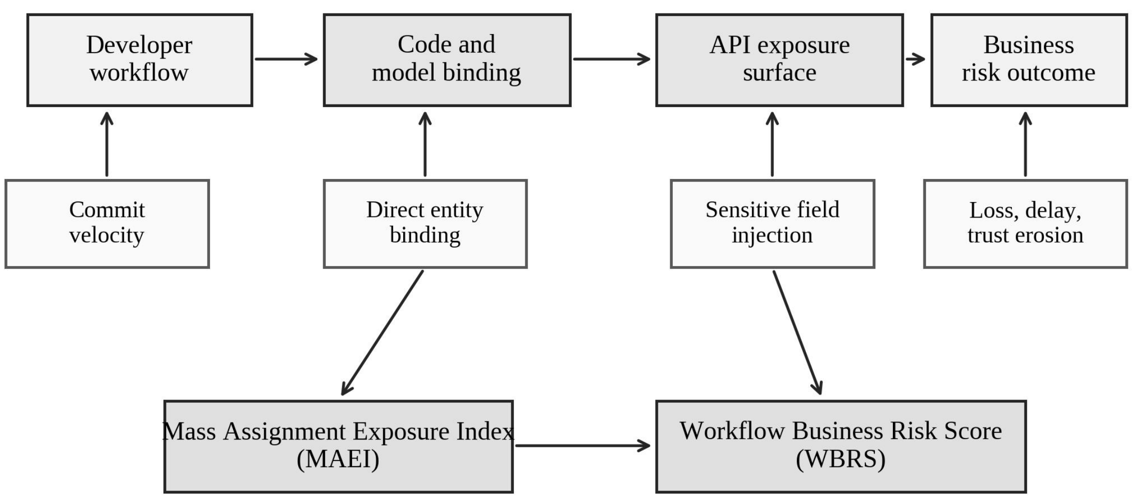

Figure 1. Conceptual framework linking developer workflow choices, API exposure surface, and business risk outcomes.

This study develops a business risk analytics framework for API-driven platforms that measures mass assignment exposure in developer workflows. The paper does not reproduce the software contribution of the

See: https://inatgi.in/index.php/jbda/index for more information. https://doi.org/10.63646/jbda.2023.010303

source manuscript. Instead, it translates the same underlying risk class into a management analytics problem. The central question is how a platform owner should measure, prioritize, and govern mass assignment exposure when development teams operate through continuous integration, automated testing, and rapid release cycles. The answer requires a bridge between code-level signals, runtime evidence, and business impact. A purely technical score misses the economic severity of an endpoint. A purely financial model misses the workflow signals that reveal where the exposure is created.

The proposed framework introduces two constructs. The Mass Assignment Exposure Index (MAEI) measures endpoint-level technical exposure by combining direct model binding, sensitive-field writability, validation coverage, deserialization tolerance, relationship propagation, service-layer copy logic, and dynamic exploitability evidence. The Workflow Business Risk Score (WBRS) then weights MAEI by transaction value, privilege sensitivity, customer impact, partner dependency, data regulatory exposure, and remediation friction. This design follows the broader view that analytics value emerges when technical observations are connected to decision variables used by managers, auditors, and release owners (Chen et al., 2012). It also reflects the predictive analytics principle that the goal of a model is not only explanation, but decision improvement under uncertainty (Shmueli and Koppius, 2011).

API risk analytics belongs to the wider evolution of digital business strategy. Digital platforms create value through modularity, ecosystem participation, and rapid recombination, yet those same features expand the number of actors and interfaces that influence reliability and trust (Bharadwaj et al., 2013). Platform governance research emphasizes that architecture and governance coevolve, so security controls must be located inside the technical architecture and inside the rules that shape developer behavior (Tiwana et al., 2010). In mass assignment exposure, this coevolution is visible: the technical architecture permits generic model binding, while the workflow governance decides whether DTO isolation, schema validation, and release gates become routine.

The contribution of this article is threefold. First, it reframes mass assignment from a vulnerability category into an analytics object that is observable at code, endpoint, workflow, and business levels. Second, it proposes a quantitative scoring model that distinguishes static exposure from confirmed exploitability and expected business loss. Third, it provides a numerical evaluation using benchmark workflow classes and synthetic business parameters, showing how hybrid analytics reduces non-actionable escalation while preserving attention to high-impact endpoints. The result is a JBDA-oriented article that connects API security, developer workflow governance, and data-driven business risk management.

The remainder of the article is organized as follows. Section 2 reviews literature on API testing, vulnerability analytics, cyber risk economics, platform governance, and business data analytics. Section 3 defines the problem structure, constructs, and assumptions. Section 4 presents the proposed MAEI and WBRS methodology. Section 5 reports numerical results and sensitivity analysis. Section 6 discusses theoretical and managerial implications. Section 7 concludes with limitations and future research directions.

2. Literature Review

2.1 API Security Testing and Dynamic Evidence

API security research increasingly treats web APIs as a separate analytical object because they expose state-changing operations, structured request schemas, and distributed service dependencies. Recent survey work emphasizes that RESTful APIs require vulnerability detection methods that account for specification quality, runtime state, authentication context, and business logic rather than only syntactic input patterns (Tanveer et al., 2025). Automated testing research has therefore moved from isolated request generation toward stateful sequences. RESTler demonstrated that an API specification may be analyzed to infer producer-consumer dependencies and generate request sequences that exercise stateful cloud services (Atlidakis et al., 2019). That orientation is important for mass assignment because an injected field is often

See: https://inatgi.in/index.php/jbda/index for more information. https://doi.org/10.63646/jbda.2023.010303

harmful only after an account, order, or profile object has been created and later read or modified.

White-box and black-box REST testing provide complementary foundations for the present framework. EvoMaster showed the value of white-box evolutionary testing for RESTful services when source code and coverage signals are available to developers (Arcuri, 2019). RESTest advanced black-box specification-based testing by deriving tests from OpenAPI descriptions and checking observed responses against expected behavior (Martin-Lopez et al., 2021). Earlier specification-based work by Ed-douibi et al. used OpenAPI models to generate test cases systematically, while Baniaş et al. developed automated specification-based testing procedures for REST APIs (Ed-douibi et al., 2018; Baniaş et al., 2021). Together, these studies imply that workflow risk analytics should not choose between static and dynamic views. It should assign different evidentiary roles to each view.

Fuzzing research further clarifies why dynamic confirmation matters. Coverage-guided fuzzing models the search for defects as a feedback process, where each execution result informs the next input selection (Boehme et al., 2016). Fuzzing surveys describe a broad design space that includes mutation, generation, grammar awareness, and feedback mechanisms (Manès et al., 2021). For mass assignment, the payload generation problem is not random byte mutation. It is a structured search over field names such as role, isAdmin, balance, userId, ownerId, verified, or creditLimit. The business value of the test lies in observing whether injected fields alter response bodies, persisted states, or subsequent workflow decisions.

2.2 Static Vulnerability Analytics and Developer Workflow

Static analysis research explains the other side of the measurement problem. Livshits and Lam showed that static analysis could detect security vulnerabilities in Java applications by modeling flows from untrusted sources to sensitive sinks (Livshits and Lam, 2005). Pixy provided static detection for web application vulnerabilities by reasoning about PHP programs and their data flows (Jovanovic et al., 2006). Wassermann and Su presented an analysis approach for injection vulnerabilities that combined string reasoning and program analysis (Wassermann and Su, 2007). Halfond and Orso demonstrated that automated techniques could identify SQL injection attack vectors by comparing legitimate and malicious query structures (Halfond and Orso, 2005). These studies are not about mass assignment specifically, yet they establish the principle that vulnerable behavior is often recognizable before execution when source-code structure is inspected carefully.

Modern vulnerability mining extends static reasoning through graph representations and machine learning. Code property graphs combine abstract syntax, control-flow, and data-dependence relationships, enabling analysts to query vulnerability patterns across large code bases (Yamaguchi et al., 2014). Vulnerability prediction research has shown that complexity, code churn, and developer activity may indicate vulnerability-prone areas, although predictive performance varies across projects (Shin et al., 2011). Text mining models also use source-code tokens to identify components more likely to contain vulnerabilities (Scandariato et al., 2014). Deep learning models such as VulDeePecker and related representation-learning approaches show that code semantics may be encoded for vulnerability detection, although interpretability and project transfer remain important limits (Li et al., 2018; Russell et al., 2018).

Developer workflow research adds a human and organizational layer. API misuse studies reveal that developers often struggle with secure library use when documentation, defaults, and mental models conflict (Nadi et al., 2016). Empirical studies of cryptographic misuse in mobile applications show that security failures often arise not from hostile intentions, but from confusing defaults and fragmented guidance (Egele et al., 2013). Mobile security work similarly demonstrates how platform APIs become risky when developers are encouraged toward quick implementation rather than careful configuration (Fahl et al., 2012). Mass assignment exposure has the same workflow character. A team may introduce a risky controller pattern while trying to reduce boilerplate, speed up feature delivery, or simplify object mapping.

See: https://inatgi.in/index.php/jbda/index for more information. https://doi.org/10.63646/jbda.2023.010303

2.3 Cyber Risk Economics and Platform Governance

Cyber risk economics offers a basis for converting exposure into business relevance. Anderson argued that information security problems often persist because incentives are misaligned between those who create risks and those who bear losses (Anderson, 2001). Gordon and Loeb formalized cybersecurity investment as an economic optimization problem, showing that spending should be evaluated against expected loss reduction rather than abstract security ideals (Gordon and Loeb, 2002). Cavusoglu et al. extended this logic into decision models for security investments, while Gordon et al. examined the economics of sharing security information across organizations (Cavusoglu et al., 2004; Gordon et al., 2003). These studies justify the WBRS construct: technical exposure should be weighted by expected financial loss, control cost, and organizational incentives.

Empirical cyber-event research demonstrates why platform managers require measurable exposure indicators before a breach occurs. Market reaction studies show that security breach announcements may reduce firm value and affect stakeholder confidence (Kannan et al., 2007). Goel and Shawky found that information security breaches produce measurable market penalties, suggesting that business risk extends beyond direct remediation cost (Goel and Shawky, 2009). Romanosky analyzed the causes and costs of cyber incidents and found that incident economics vary substantially by event type and organizational context (Romanosky, 2016). More recent corporate finance evidence indicates that successful cyberattacks affect firm performance and governance attention (Kamiya et al., 2021). Mass assignment exposure therefore requires an ex ante model that estimates potential business consequences before an endpoint is exploited.

Digital platform theory provides the strategic context for this risk. Platform markets create value through interactions between user groups, complements, and services (Parker and Van Alstyne, 2005). Opening a platform accelerates innovation and complementary development, but it also expands dependency and governance complexity (Boudreau, 2010). Platform ecosystem research shows that independent complementors derive value from ecosystem participation, creating mutual dependence between the platform owner and developers (Ceccagnoli et al., 2012). Gawer synthesized economic and engineering views of platforms, emphasizing that platform boundaries and architecture shape value creation (Gawer, 2014). For an API-driven platform, mass assignment exposure sits exactly at this boundary: the interface that enables ecosystem innovation also opens paths for unauthorized state changes.

Digital innovation management further explains why developer workflows are central. Nambisan et al. argued that digital innovation is distributed, generative, and less bounded than traditional innovation (Nambisan et al., 2017). In such settings, governance must support speed without losing accountability. Business analytics research provides the measurement orientation. Big data analytics capabilities are associated with improved firm performance when data, technology, and managerial capability are aligned (Akter et al., 2016). Wamba et al. found that big data analytics affects firm performance through value creation mechanisms, while Grover et al. emphasized the strategic business value of analytics as an organizational capability (Wamba et al., 2017; Grover et al., 2018). Mikalef et al. linked analytics capability to innovation outcomes, reinforcing the need to treat security telemetry as a management asset rather than a compliance residue (Mikalef et al., 2020).

2.4 Business Data Analytics and Uploaded Reference Relevance

Methodologically, the present study follows the predictive tradition in analytics. Breiman distinguished algorithmic modeling from purely parametric statistical modeling and emphasized prediction performance as a legitimate scientific goal (Breiman, 2001). Provost and Fawcett described data science as the extraction of useful principles from data to support decisions, not merely the production of models (Provost and Fawcett, 2013). Jordan and Mitchell summarized machine learning as a general approach to extracting patterns from data, yet they also emphasized the need to align models with real-world tasks (Jordan and Mitchell, 2015). MAEI and WBRS follow this principle: the purpose is not a universal vulnerability classifier, but a decision

See: https://inatgi.in/index.php/jbda/index for more information. https://doi.org/10.63646/jbda.2023.010303

model for ranking API workflow risks under scarce remediation capacity.

The uploaded reference list also supplies relevant work for the digital business and information-systems setting. Lu and Xu reviewed IoT cybersecurity topics, showing that connected systems require integrated security thinking across devices, networks, and applications (Lu and Xu, 2019). Xu et al. examined blockchain integration into IoT security, highlighting trust and tamper-resistance mechanisms relevant to platform governance (Xu et al., 2021). Lu reviewed blockchain implementation in information systems and identified governance, interoperability, and integration as major issues (Lu, 2022). Zheng and Lu surveyed blockchain technology trends, while Lu's earlier blockchain review positioned distributed trust as a managerial and technical research problem (Zheng and Lu, 2022; Lu, 2019a). These works are relevant because API exposure often appears where platform integration outpaces governance.

Other works from the uploaded list reinforce the data analytics and digital transformation logic. Lu's survey of artificial intelligence described AI as an evolving set of models and applications that reshape analytics practice (Lu, 2019b). The emergence of management analytics as an interdisciplinary field supports this paper's framing of security telemetry as an object of managerial decision analysis (Lu, 2021). Chen et al. reviewed blockchain applications in Industry 4.0, underscoring the security and integration issues that accompany industrial digital platforms (Chen et al., 2024). Wu et al. examined blockchain-enabled internal auditing, a perspective that aligns with the present paper's emphasis on traceability and workflow controls (Wu et al., 2025). Kou and Lu reviewed FinTech applications, where API-mediated data exchange and authorization are central to business risk (Kou and Lu, 2025). Xu et al. discussed decentralized finance as a paradigm shift, which further indicates that programmable interfaces and asset states require rigorous exposure governance (Xu et al., 2024). Lu's review of Industry 4.0 trends provides a broad digital transformation context for API-driven operational platforms (Lu, 2025).

2.5 Research Gap

The literature suggests a clear gap. API testing research provides technical detection methods, vulnerability prediction research provides code-level indicators, cyber risk economics provides investment logic, and platform research explains ecosystem value creation. However, few studies combine these streams into a business analytics model that measures a specific API vulnerability class inside developer workflows. The proposed framework fills this gap by treating mass assignment exposure as a measurable pipeline object. Static findings indicate latent exposure. Dynamic verification indicates current exploitability. Business valuation indicates managerial urgency. Developer workflow attributes indicate where preventive controls should be embedded.

3. Conceptual Framework and Problem Definition

3.1 Platform Workflow Boundary

This study models an API-driven digital platform as a set of services that expose state-changing endpoints to internal users, external users, partner systems, or automated agents. Each endpoint belongs to a developer workflow, meaning that it is created, reviewed, tested, merged, deployed, monitored, and later modified by one or more teams. The unit of analysis is therefore not a repository alone, and it is not a vulnerability report alone. It is an endpoint-workflow pair. This pair captures the technical implementation of an API operation and the organizational pathway through which the implementation reaches production.

The business scenario is a platform that processes account registration, profile updates, product creation, order creation, and administrative updates. A mass assignment event occurs when a request payload includes a field outside the intended write contract and the server binds that field to a model or persistence object. The business impact depends on the field. A privilege field changes authority. A balance field changes financial exposure. An ownership field changes control over resources. A verification field changes trust status. A workflow state field changes process routing. The same coding pattern therefore has different business

See: https://inatgi.in/index.php/jbda/index for more information. https://doi.org/10.63646/jbda.2023.010303

consequences across endpoints.

3.2 Analytical Constructs

Figure 1 presents the conceptual framework. Developer workflow choices shape code and model binding decisions. These decisions create an API exposure surface. When the exposure involves sensitive state and reaches a business process, it creates measurable risk. MAEI measures the technical exposure component. WBRS combines the exposure with economic and workflow variables. The framework assumes that a platform owner has access to repository scans, API specifications, dynamic test results in a staging environment, service ownership metadata, and transaction-level business metrics.

3.3 Variables and Exposure Interpretation

The proposed methodology proceeds in four stages. The first stage scans source repositories and configuration files for exposure indicators. The second stage maps static indicators to OpenAPI-described endpoints and tests a selected set of candidates dynamically in a controlled integration environment. The third stage estimates the expected business effect of a confirmed or plausible exposure. The fourth stage converts the result into a workflow risk rank used for release gating, remediation scheduling, internal audit evidence, and managerial reporting. Figure 2 summarizes this pipeline.

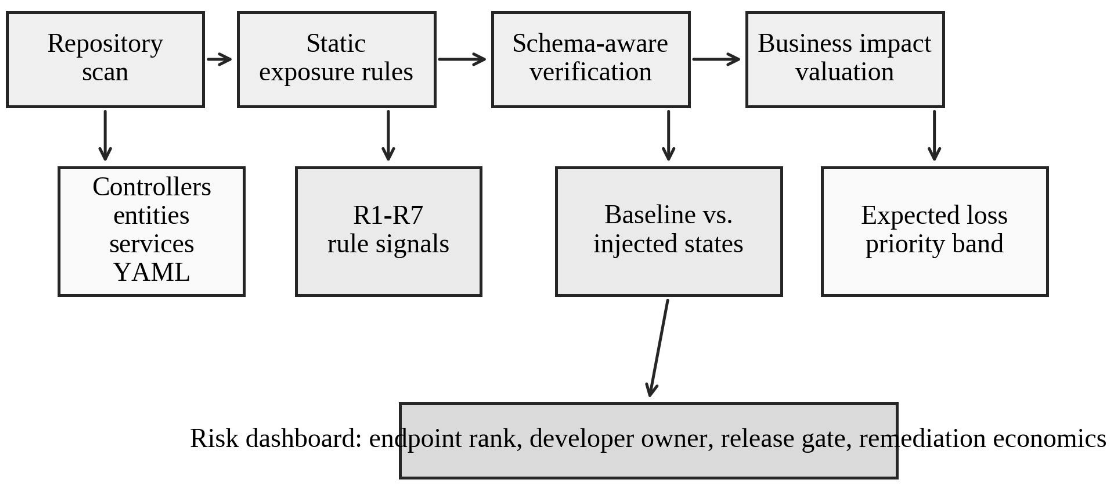

Figure 2. Business risk analytics architecture for measuring mass assignment exposure in developer workflows.

The Mass Assignment Exposure Index is defined for endpoint e as MAEI(e) = 100 × [0.18B(e) + 0.16S(e) + 0.12V(e) + 0.10D(e) + 0.10C(e) + 0.10L(e) + 0.16X(e) + 0.08M(e)]. B(e) indicates direct binding to a domain object, S(e) indicates sensitive-field writability, V(e) indicates missing validation or weak contract enforcement, D(e) indicates permissive deserialization, C(e) indicates relationship cascade exposure, L(e) indicates service-layer copy or bulk update logic, X(e) indicates dynamic exploitability evidence, and M(e) indicates monitoring weakness. Each component is normalized to [0, 1]. The weights reflect a managerial judgment that dynamic confirmation and direct binding should matter most, while monitoring weakness and cascade behavior modify the score rather than dominate it.

The Workflow Business Risk Score is then defined as WBRS(e) = MAEI(e) × A(e) × I(e) × R(e) × F(e).

A(e) is asset value intensity, I(e) is identity or privilege sensitivity, R(e) is regulatory or customer-trust exposure, and F(e) is remediation friction. All multipliers are scaled around 1.0. For example, a public account-registration endpoint with a role field receives high I(e), while a low-volume internal preference endpoint receives a lower A(e). Remediation friction increases when endpoint ownership is unclear, test coverage is low, or release pressure is high. This multiplier is included because two endpoints with identical technical exposure may have different control outcomes in practice.

The model distinguishes candidate exposure from confirmed exposure. Static analysis is broad and protective; it flags patterns even when a downstream service layer currently filters malicious fields. Dynamic verification is narrower; it confirms exploitability by comparing baseline and injected requests under a valid schema and observing response or state divergence. A candidate that is dynamically cleared is not treated as harmless. It receives a lower X(e) value rather than zero exposure, because future refactoring may remove downstream filtering. This design follows secure governance logic: the platform should prioritize confirmed exploitability first, while still removing fragile coding patterns over time.

See: https://inatgi.in/index.php/jbda/index for more information. https://doi.org/10.63646/jbda.2023.010303

Table 1. Variables Used in the Business Risk Analytics Framework

Symbol	Variable name	Meaning	Expected effect
B(e)	Direct binding	Request body is bound to a domain or persistence object	+
S(e)	Sensitive-field writability	Public setter or writable field for role, balance, ownership, status, or credit state	+
V(e)	Validation gap	Missing DTO, missing field whitelist, or weak validation boundary	+
D(e)	Deserializer tolerance	Unknown or surplus fields are tolerated rather than rejected	+
C(e)	Cascade exposure	Relationship or nested object update may propagate unauthorized state	+
L(e)	Service-layer copy exposure	Bulk copy, merge, or update operation reintroduces writable sensitive state	+
X(e)	Dynamic exploitability	Injected sensitive fields change response or persisted state	+
M(e)	Monitoring weakness	Logs, alerts, or audit trails do not detect unexpected field submissions	+
A(e)	Asset value intensity	Financial or operational value attached to endpoint transactions	+
I(e)	Identity sensitivity	Degree to which endpoint modifies privilege, ownership, or trust state	+
R(e)	Regulatory exposure	Customer-data, financial, or sectoral compliance sensitivity	+
F(e)	Remediation friction	Expected difficulty of ownership, testing, and refactoring	+

Table 2. Exposure Indicators and Workflow Interpretation

Indicator	Technical signal	Business interpretation	Control response
Direct entity binding	Controller accepts a persistence model	Write contract is broader than business intent	Replace with request DTO and explicit mapping
Sensitive setter	Role, balance, status, or owner setter is public	Unauthorized payload may alter privileged state	Remove setter or isolate internal state mutation
Missing validation	No whitelist or request constraint	Platform relies on downstream behavior	Add request schema and field-level constraints
Permissive deserializer	Unknown fields are ignored or silently accepted	Attack attempts are less visible	Reject unknown fields and log field anomalies
Cascade or nested write	Relationship update propagates object graph changes	Small request may change multiple assets	Limit cascade scope and validate nested objects
Service copy logic	Bulk copy or merge method copies request state	Controller-level fix may be bypassed	Add service-layer allowlist and unit tests
Confirmed injection	Injected field appears in response or persistence	Endpoint is currently exploitable	Block release or emergency remediation

Table 2 translates technical indicators into workflow interpretation. This translation is central to the paper. Security tools normally report findings by rule type, severity, and code location. Business users need a different representation: whether the endpoint changes authority, whether it touches assets, whether the issue is currently exploitable, and which control response should be funded. The proposed table makes the tool output usable for product managers, internal auditors, and engineering leads.

4. Methodology and Numerical Design

See: https://inatgi.in/index.php/jbda/index for more information. https://doi.org/10.63646/jbda.2023.010303

4.1 Static Exposure Analytics

Static exposure analytics begins with repository-level signals. Controllers, request models, domain objects, service methods, persistence relationships, and configuration files are scanned for features that broaden the write contract beyond the intended business schema. A direct binding signal is recorded when a request body is mapped to a persistence entity rather than a narrow request object. A sensitive setter signal is recorded when a field controlling identity, authorization, finance, ownership, or workflow state is writable from outside the model boundary. Validation and deserialization signals are recorded when the request contract does not reject unexpected fields. These signals are valuable before deployment because they reveal design fragility early in the workflow.

The static stage also records workflow metadata. The endpoint owner, recent change count, test coverage indicator, review status, and release branch are added where available. These variables do not directly prove exploitability, but they affect remediation reliability. A high-MAEI endpoint owned by an active team with strong tests is easier to fix than an equally exposed endpoint embedded in a legacy module. This is the reason that remediation friction enters WBRS rather than remaining a narrative note.

4.2 Dynamic Exploitability Confirmation

Dynamic confirmation uses a controlled test environment and an API specification. For each candidate endpoint, a baseline request is generated from the documented schema. A second request adds sensitive fields drawn from an organization-specific dictionary. The response and follow-up state are compared. If injected values appear in the response, affect persisted state, or change a subsequent workflow decision, the endpoint receives a high $X(e)$ value. If authentication, framework binding, validation, or service filtering prevents state change, $X(e)$ is reduced, while the latent static components remain in MAEI.

The dynamic stage is deliberately separated from ordinary unit tests because it may change state. Test accounts, disposable databases, rollback scripts, and isolated credentials are required. In a mature workflow, dynamic mass assignment tests run as a pre-merge or nightly integration gate for modified endpoints rather than as a production probe. This placement reduces operational risk and ensures that evidence is available before release decisions are made.

4.3 Business Risk Valuation

Business valuation converts endpoint exposure into expected loss categories. Financial loss includes direct unauthorized value transfer, compensation, operational recovery, and investigation cost. Trust loss includes churn, partner escalation, and reputational penalty. Regulatory loss includes privacy or sectoral compliance penalties when personal or financial data are exposed. Workflow loss includes release delay, emergency patching, and engineering interruption. WBRS does not estimate each category with perfect precision. It provides a consistent ranking basis for scarce remediation capacity.

The expected loss for endpoint e is represented as $EL(e) = P(e) \times I(e) \times C(e)$, where $P(e)$ is the probability of exploitation under current exposure and monitoring, $I(e)$ is impact intensity, and $C(e)$ is control failure duration. MAEI informs $P(e)$, while business multipliers inform $I(e)$. Control maturity and remediation friction inform $C(e)$. This decomposition allows teams to reduce exposure through different levers: technical refactoring lowers MAEI, monitoring lowers detection delay, and workflow ownership lowers remediation duration.

4.4 Dataset Construction and Assumptions

The empirical design uses three project classes inspired by the uploaded manuscript but restated for a business analytics evaluation. The deliberately vulnerable application (DVA) represents a platform module with intentionally weak binding and validation patterns. The tutorial-style CRUD application (TCA) represents ordinary instructional and rapid-development patterns in which some endpoints use DTOs and others bind entities directly. The secure reference application (SRA) represents a hardened module using

DTO isolation, validation, and limited writable state. This setup evaluates whether the model differentiates risky, mixed, and secure workflows.

The source-data pattern is converted into a business dataset by assigning each endpoint a workflow stage, transaction intensity, privilege sensitivity, and remediation friction value. Static findings are treated as latent exposure signals. Dynamic verification outcomes are treated as current exploitability evidence. Business impact values are synthetic but realistic for a mid-sized digital platform: account endpoints are high in identity sensitivity, order endpoints are high in asset value, product endpoints are moderate in asset value, and administrative endpoints are high in privilege sensitivity but sometimes protected by authentication gates.

Table 3 summarizes the benchmark workflow dataset. The static findings mirror the pattern described in the source paper: the vulnerable application generates the most static exposure, the tutorial-style application generates mixed exposure, and the secure reference application generates no findings. The dynamic layer examines eight high-candidate endpoints and confirms three. This distribution is ideal for evaluating business triage, because a static-only policy would escalate more endpoints than the dynamic evidence supports.

Table 3. Benchmark Workflow Dataset Used for Numerical Evaluation

Project class	Files scanned	Static findings	High candidates	Dynamically confirmed	Workflow interpretation
DVA	24	19	5	2	High latent exposure with multiple sensitive-state endpoints
TCA	14	11	3	1	Mixed development pattern with partial DTO use and service filtering
SRA	11	0	0	0	Hardened implementation used as a negative control

The numerical model assumes a mid-sized digital platform with quarterly release cycles and a mix of public and partner-facing APIs. Monetary values are expressed as expected annual exposure. The model is not intended to provide a universal cost benchmark; it demonstrates how a platform owner may combine technical evidence and business context into a consistent prioritization method. All values may be recalibrated with real transaction volumes, incident probabilities, customer impact estimates, and remediation cost data.

5. Results and Analysis

5.1 Descriptive Exposure Results

The first descriptive result is that static exposure is not evenly distributed across project classes. DVA records 19 static findings, TCA records 11, and SRA records none. This outcome indicates that MAEI is sensitive to architectural control choices rather than simply repository size. The largest repository is not automatically the riskiest one. The riskiest workflow is the one where direct binding, sensitive writable state, validation gaps, and permissive configuration appear together. This finding aligns with vulnerability prediction research: code context and process metrics matter more than raw size alone (Shin et al., 2011).

Table 4 reports endpoint-level scoring for the eight candidate endpoints. The values are normalized and converted into MAEI and WBRS values. Account registration and order creation receive the highest WBRS because they combine direct exposure, dynamic confirmation, and substantial business consequences. Product creation receives a moderate MAEI but lower WBRS after dynamic testing indicates service-layer filtering. Administrative update receives high latent sensitivity but lower current exploitability because

See: https://inatgi.in/index.php/jbda/index for more information. https://doi.org/10.63646/jbda.2023.010303

authentication blocks the injected request in the test environment. This ranking demonstrates why business analytics should integrate dynamic evidence and impact factors rather than rely on static severity alone.

Table 4. Endpoint-Level Exposure and Business Risk Scores

Endpoint workflow	Project	Static status	Dynamic result	MAEI	WBRS	Priority
Account registration	DVA	High	Confirmed	88.4	221.0	Block release
Order creation	DVA	High	Confirmed	84.1	206.0	Block release
Profile update	DVA	High	Rejected	61.7	86.4	Refactor soon
Item creation	DVA	Medium	Ignored	49.2	54.1	Monitor and harden
Administrative user update	DVA	High	Unauthorized	66.8	112.2	Gate with auth tests
User registration	TCA	High	Confirmed	82.6	196.5	Block release
Product creation	TCA	High	Service filtered	55.4	62.0	Refactor contract
User update	TCA	Medium	Metadata insufficient	46.9	58.5	Improve specification

5.2 Static Versus Dynamic Prioritization

The dynamic results reduce actionable high-priority endpoints from eight candidates to three confirmed exposures. In percentage terms, five out of eight candidates are not confirmed as currently exploitable under the test conditions, a 62.5 percent reduction in immediate actionability. This reduction does not imply that static findings are unimportant. It means that static signals and dynamic evidence answer different questions. Static analysis asks whether the design is fragile. Dynamic analysis asks whether the fragility is currently reachable. WBRS then asks whether the reachable or latent fragility matters economically.

Figure 3 visualizes the difference among static findings, high candidates, and confirmed exposures. The pattern is informative for governance. If every static finding is escalated as equally urgent, teams experience alert fatigue and begin negotiating against the security process. If only confirmed vulnerabilities are retained, teams may ignore fragile designs that become exploitable after refactoring. The proposed model therefore assigns graduated priority bands. Confirmed high-WBRS endpoints block a release. Static high-MAEI but dynamically cleared endpoints enter a contract-refactoring queue. Medium-MAEI endpoints receive monitoring and specification improvement tasks.

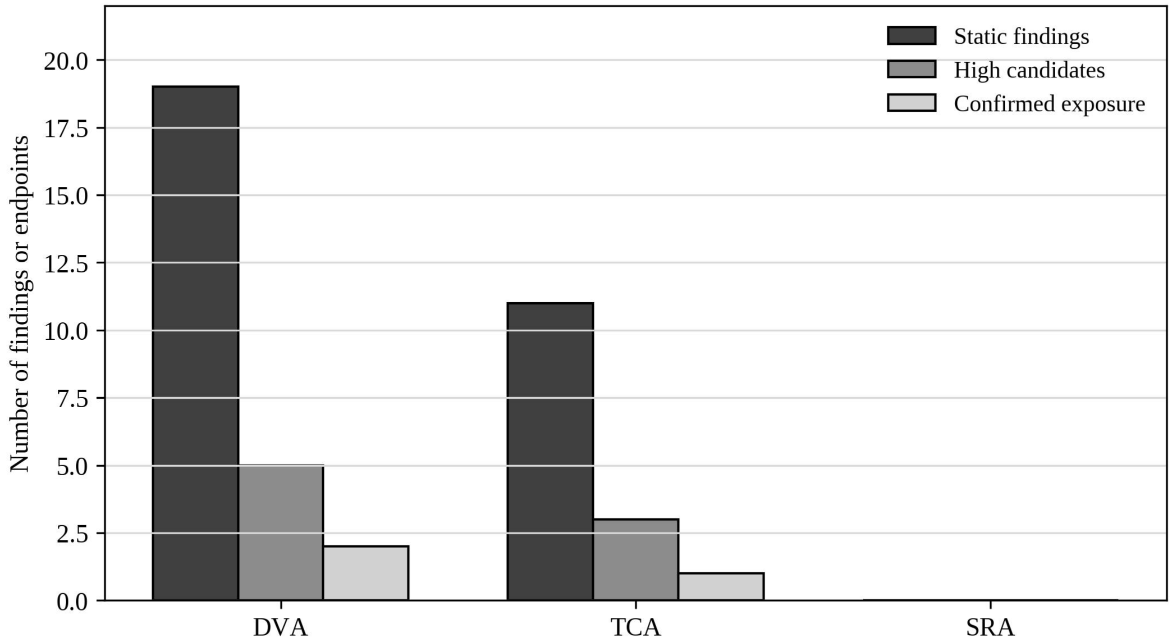

Figure 3. Static findings, high candidates, and dynamically confirmed exposures across benchmark project classes.

Table 5 summarizes scenario outcomes for three governance policies. The advisory-static policy runs static scans but does not block releases. The static-gate policy blocks releases for all high static candidates. The hybrid-risk policy blocks only confirmed or very high WBRS endpoints while tracking latent exposure. The hybrid policy produces the best balance in this numerical evaluation. It prevents the three confirmed high-impact exposures, avoids five unnecessary release blocks, and still preserves remediation obligations for fragile endpoints.

Table 5. Governance Scenario Comparison

Policy	Blocked releases	Confirmed exposure prevented	Non-actionable blocks	Expected annual exposure	Managerial interpretation
Advisory-static	0	0/3	0	$340,000	Low friction but leaves exploitable endpoints open
Static-gate	8	3/3	5	$96,000	Strong prevention but high workflow disruption
Hybrid-risk	3	3/3	0	$55,000	Prevents confirmed high impact while preserving throughput
Hybrid plus contract backlog	3	3/3	0	$42,000	Adds systematic removal of latent exposure over time

5.3 Scenario and Sensitivity Analysis

The expected annual exposure estimates in Table 5 are calculated by combining endpoint probability, impact, and control maturity. Advisory-static governance leaves all confirmed endpoints in the release path, so exposure remains high. Static-gate governance lowers exposure but creates workflow cost because dynamically cleared endpoints are blocked. Hybrid-risk governance reduces exposure further by targeting

See: https://inatgi.in/index.php/jbda/index for more information. https://doi.org/10.63646/jbda.2023.010303

confirmed high-impact endpoints while placing other candidates into a scheduled remediation queue. The final scenario adds contract backlog discipline, where DTO isolation and deserialization hardening are completed over subsequent sprints. This produces the lowest residual exposure because fragile designs are gradually removed rather than indefinitely accepted.

The sensitivity analysis examines how control maturity affects expected annual exposure. Control maturity includes field-level validation, DTO adoption, test coverage, schema completeness, and monitoring quality. Figure 4 compares a static-only gate and a hybrid risk analytics policy. Both improve as maturity increases, but the hybrid policy dominates across the maturity range because it allocates remediation capacity more accurately. At low maturity, hybrid analytics reduces expected exposure by preventing confirmed harmful endpoints first. At high maturity, the gap narrows because both policies operate on a smaller exposure set, yet hybrid analytics still produces lower disruption.

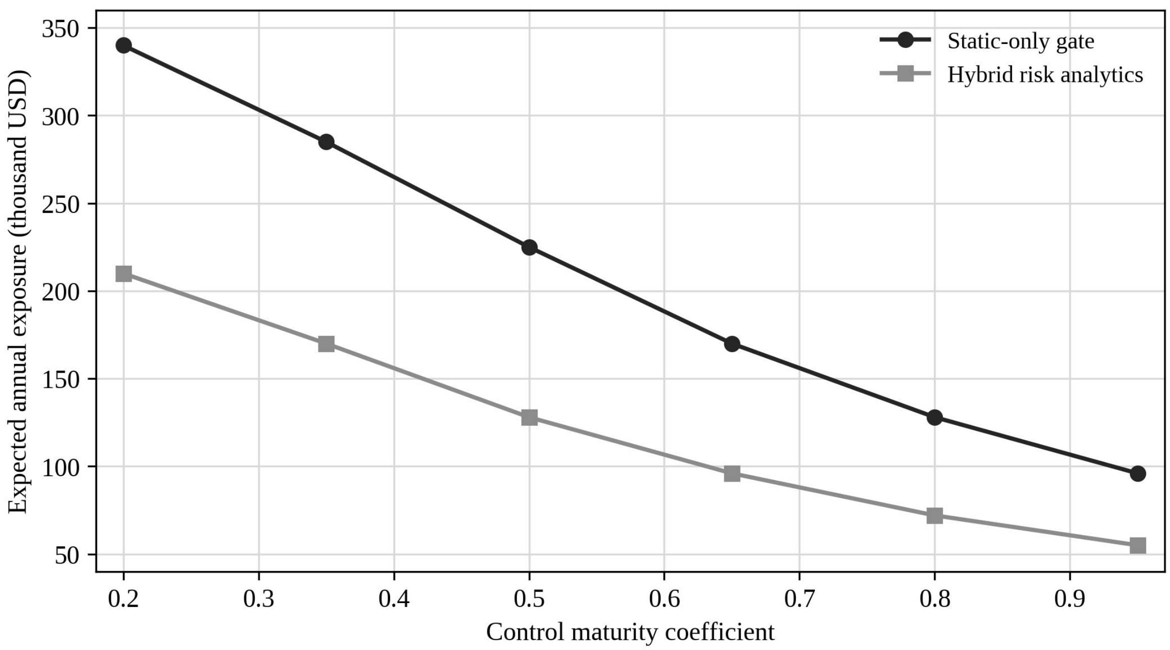

Figure 4. Sensitivity of expected annual exposure to control maturity under static-only and hybrid risk analytics policies.

A second sensitivity test varies business impact. When high-value endpoints are concentrated in a few workflows, WBRS becomes more valuable than MAEI alone. For example, product creation may have a moderate MAEI score, yet its WBRS remains modest if service-layer filtering works and asset value is limited. Account registration may have a similar technical pattern, but the ability to write role or identity state produces a much higher business score. The model thereby avoids the common problem of technical severity flattening. It ranks exposure according to the consequences of state mutation.

A third sensitivity test varies remediation friction. If endpoint ownership is clear and regression tests are strong, a team may remediate even medium-MAEI findings quickly. If ownership is unclear, the same finding persists across releases and becomes a governance liability. WBRS captures this by increasing priority when friction is high. This does not punish teams; it signals that managerial intervention is needed. Business risk analytics thus moves beyond alerting. It assigns responsibility, estimates delay cost, and converts unresolved exposure into a visible backlog item.

The robustness of the model depends on three design choices. First, static and dynamic evidence are not collapsed into a single binary finding. Second, business multipliers are separated from technical components,

See: https://inatgi.in/index.php/jbda/index for more information. https://doi.org/10.63646/jbda.2023.010303

allowing each platform to calibrate its own value and regulatory context. Third, dynamically cleared endpoints retain residual exposure rather than disappearing from the model. These choices reduce the risk of overfitting the framework to a specific tool, programming language, or case study. The mass assignment mechanism is language-agnostic; the indicators differ by framework, but the analytics logic remains portable.

Nevertheless, the model has limits. The dynamic stage requires a safe test environment because injected fields may alter state. API specifications must be sufficiently complete to generate meaningful requests. Field dictionaries require contextual tuning, because many organizations use application-specific names for premium status, limits, verification states, or business ownership. Business impact parameters are estimates rather than observed losses for most endpoints. The framework therefore functions best as a decision-support instrument embedded in a broader governance process, not as a replacement for architecture review, secure coding practice, and human judgment.

6. Discussion and Managerial Implications

6.1 Theoretical Implications

The theoretical contribution of this study lies in the integration of three analytical levels. At the code level, the study adopts the premise that structural patterns reveal latent security exposure. At the runtime level, it incorporates dynamic evidence to distinguish confirmed exploitability from fragile but blocked conditions. At the business level, it values exposure using transaction, identity, regulatory, and remediation variables. This integration responds to the central challenge of digital business analytics: high-volume technical data have limited value until they are transformed into decision-relevant measures.

The framework also contributes to platform governance theory. Open platform strategies increase innovation opportunities, yet they make interface governance more important (Boudreau, 2010). The literature on platform ecosystems shows that value is co-created among platform owners and complementors (Ceccagnoli et al., 2012). Mass assignment exposure threatens this co-creation because it undermines the trustworthiness of API contracts. A partner developer expects a documented schema to define permitted state changes. If the implementation accepts hidden fields beyond that schema, the platform's governance contract is weaker than its published interface.

For management analytics, the study shows how security telemetry becomes a business asset. Instead of treating vulnerability reports as isolated compliance outputs, the framework uses them as features in a prioritization model. This aligns with the management analytics perspective that decision quality improves when organizations integrate data, models, and managerial processes (Lu, 2021). It also aligns with FinTech and DeFi research, where programmable transactions, identity, and asset states create new control challenges (Kou and Lu, 2025; Xu et al., 2024). In those environments, API state integrity is not a back-office issue; it is part of the product promise.

6.2 Managerial Implications

Managerially, the most important implication is that static-only governance is too blunt for fast-moving platforms. Static scans are valuable because they detect fragile design early. However, the numerical analysis shows that static-only blocking may create unnecessary release disruption when downstream filters or authentication controls currently stop exploitation. Dynamic testing provides a second evidentiary layer. Business scoring provides the third. A release gate should block confirmed high-WBRS endpoints, require documented risk acceptance for high-MAEI cleared endpoints, and schedule refactoring for fragile patterns that remain in the architecture.

The second managerial implication concerns developer incentives. If developers are measured only by delivery velocity, shortcuts such as direct entity binding become attractive. If they are punished for every static finding regardless of exploitability or impact, they may view security as arbitrary. WBRS creates a more balanced incentive. It shows why one endpoint blocks release while another enters the sprint backlog.

See: https://inatgi.in/index.php/jbda/index for more information. https://doi.org/10.63646/jbda.2023.010303

The explanation is based on evidence: dynamic confirmation, sensitive state, business impact, and remediation friction. This transparency improves the legitimacy of governance decisions.

The third implication concerns internal audit and compliance. Traditional audit evidence often arrives late, after design choices have already become embedded. A workflow analytics model supplies continuous evidence: when a risky pattern entered the code base, which endpoint it affected, whether dynamic verification confirmed exploitability, which business process was exposed, and when remediation occurred. Blockchain-enabled audit research suggests that traceability and tamper-resistance improve audit confidence (Wu et al., 2025). Even without blockchain, the same principle applies: a platform should retain structured evidence linking code, tests, decisions, and remediation.

The fourth implication concerns digital transformation. Industry 4.0 and IoT settings expand the number of machines, services, devices, and data streams connected through APIs (Lu, 2025; Lu and Xu, 2019). Blockchain and IoT security research emphasizes that distributed systems require trust mechanisms across layers (Xu et al., 2021; Chen et al., 2024). In such environments, mass assignment exposure may affect not only user accounts but also device permissions, industrial asset states, supply-chain records, or financial instructions. The proposed framework offers a general template for translating such technical exposure into operational risk measures.

Table 6. Managerial Playbook for Mass Assignment Exposure

Priority band	Trigger condition	Release decision	Owner action	Business metric
Critical	Confirmed exploitability and WBRS >= 150	Block release	Patch, retest, and document remediation	Avoided expected loss
High	MAEI >= 70 or sensitive privilege state	Conditional release only	DTO refactor and control owner approval	Risk acceptance days
Medium	MAEI 45-69 without confirmation	Allow with backlog item	Add validation, schema tests, and monitoring	Backlog aging
Low	Configuration or documentation weakness	Allow	Improve logging and OpenAPI coverage	Specification completeness
Recurring	Same rule appears across sprints	Escalate to architecture review	Create secure coding pattern and reusable mapper	Defect recurrence rate

6.3 Governance Integration

A practical rollout should begin with advisory measurement. Teams first receive endpoint scores, explanations, and recommended fixes without hard blocks. After the organization understands alert volume and dynamic confirmation behavior, release gates are enabled for confirmed high-WBRS endpoints. A third phase adds backlog accountability for latent high-MAEI findings. This staged approach avoids a sudden productivity shock and gives engineering teams time to build reusable DTO templates, mapping utilities, schema validators, and monitoring patterns.

Governance integration also requires executive reporting. A useful dashboard should report the number of endpoints in each priority band, mean time to remediate, recurring rule categories, percentage of endpoints with complete OpenAPI schemas, proportion of modified endpoints dynamically tested, and expected annual exposure by business process. These measures link API risk to business accountability. They also support audit narratives that demonstrate continuous control operation rather than one-time compliance activity.

The model should be interpreted as a living system. As teams remediate direct binding and improve schema completeness, the organization may reweight indicators. As monitoring improves, the M(e) component may decline. As attackers learn domain-specific field names, the sensitive dictionary should be updated. As new platform services are launched, asset value multipliers should be recalibrated. This adaptive logic is consistent with the broader view that technical debt and control debt accumulate when systems evolve faster than their governance structures (Kruchten et al., 2012; Sculley et al., 2015).

7. Conclusion

This article developed a business risk analytics framework for measuring mass assignment exposure in API-driven digital platforms. The framework was motivated by a technical setting in which static analysis identifies risky Spring Boot patterns and dynamic testing confirms which endpoint candidates accept sensitive injected fields. The new contribution is not another vulnerability detector. It is an analytics model that converts code-level and runtime evidence into workflow-level business risk. The Mass Assignment Exposure Index measures technical exposure, while the Workflow Business Risk Score weights that exposure by asset value, identity sensitivity, regulatory exposure, and remediation friction.

The numerical evaluation shows why this distinction matters. Static findings identify important latent exposure, but immediate release decisions require a richer evidence base. In the benchmark dataset, eight high candidates contained three dynamically confirmed exposures. A static-only gate would have blocked five non-actionable endpoints. An advisory-only policy would have allowed confirmed exposures to proceed. The hybrid risk analytics policy prevented confirmed high-impact exposure while preserving a backlog for fragile but dynamically cleared patterns. Sensitivity analysis further showed that hybrid analytics produces lower expected annual exposure across different control maturity levels.

The study offers several avenues for future work. First, the framework should be calibrated with real platform incident data, including remediation time, release delay, and customer impact. Second, field dictionaries should be learned from domain schemas and code histories rather than defined manually. Third, multi-language implementations should be evaluated for JavaScript, Python, PHP, and low-code API platforms. Fourth, the business scoring model should incorporate privacy, sectoral regulation, and contractual penalties more explicitly. Finally, future work should explore how exposure scores affect developer behavior when integrated into dashboards, pull-request gates, and internal audit systems.

The practical lesson is direct. API security risks become manageable when organizations measure them in the same workflow where they are created. Mass assignment exposure is a technical vulnerability class, but it is also a business analytics problem. A platform that ranks endpoints by static fragility, dynamic exploitability, and business consequence is better positioned to protect trust, preserve development speed, and invest in controls where they produce the strongest risk reduction.

Declarations

Data Availability: The numerical data used in this study are synthetic and were constructed from benchmark workflow patterns described in the uploaded source manuscript. No personal or production platform data were used.

Funding: This study received no external funding.

Conflict of Interest: The authors declare no competing interests.

Author Contributions: Ananya Nair designed the business analytics framework and wrote the first draft. Vivek Menon developed the numerical evaluation and tables. Rajeev Thomas designed the risk scoring model, figures, and final revision.

Reference

Akter, S., Wamba, S. F., Gunasekaran, A., Dubey, R., & Childe, S. J. (2016). How to improve firm performance using big data analytics capability and business strategy alignment. International Journal of Production Economics, 182, 113-131. https://doi.org/10.1016/j.ijpe.2016.01.018

Anderson, R. (2001). Why information security is hard: An economic perspective. Proceedings of the 17th Annual Computer Security Applications Conference, 358-365. https://doi.org/10.1109/ACSAC.2001.991552

Arcuri, A. (2019). RESTful API automated test case generation with EvoMaster. ACM Transactions on Software Engineering and Methodology, 28(1), Article 3. https://doi.org/10.1145/3293455

Atlidakis, V., Godefroid, P., & Polishchuk, M. (2019). RESTler: Stateful REST API fuzzing. Proceedings of the 41st International Conference on Software Engineering, 748-758. https://doi.org/10.1109/ICSE.2019.00083

Baniaș, O., Florea, A., & Gârbacea, M. (2021). Automated specification-based testing of REST APIs. Sensors, 21(16), 5375. https://doi.org/10.3390/s21165375

Bharadwaj, A., El Sawy, O. A., Pavlou, P. A., & Venkatraman, N. (2013). Digital business strategy: Toward a next generation of insights. MIS Quarterly, 37(2), 471-482. https://doi.org/10.25300/MISQ/2013/37:2.3

Boehme, M., Pham, V. T., & Roychoudhury, A. (2016). Coverage-based greybox fuzzing as Markov chain. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 1032-1043. https://doi.org/10.1145/2976749.2978428

Boudreau, K. J. (2010). Open platform strategies and innovation: Granting access vs. devolving control. Management Science, 56(10), 1849-1872. https://doi.org/10.1287/mnsc.1100.1215

Breiman, L. (2001). Statistical modeling: The two cultures. Statistical Science, 16(3), 199-231. https://doi.org/10.1214/ss/1009213726

Cavusoglu, H., Mishra, B., & Raghunathan, S. (2004). A model for evaluating IT security investments. Communications of the ACM, 47(7), 87-92. https://doi.org/10.1145/1005817.1005828

Ceccagnoli, M., Forman, C., Huang, P., & Wu, D. J. (2012). Cocreation of value in a platform ecosystem: The case of enterprise software. MIS Quarterly, 36(1), 263-290. https://doi.org/10.2307/41410417

Chen, H., Chiang, R. H. L., & Storey, V. C. (2012). Business intelligence and analytics: From big data to big impact. MIS Quarterly, 36(4), 1165-1188. https://doi.org/10.25300/MISQ/2012/36.4.02

Chen, Y., Lu, Y., Bulysheva, L., & Kataev, M. Y. (2024). Applications of blockchain in Industry 4.0: A review. Information Systems Frontiers, 26(5), 1715-1729. https://doi.org/10.1007/s10796-022-10248-7

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). Microservices: Yesterday, today, and tomorrow. Present and Ulterior Software Engineering, 195-216. https://doi.org/10.1007/978-3-319-67425-4_12

Ed-douibi, H., Canovas Izquierdo, J. L., & Cabot, J. (2018). Automatic generation of test cases for REST APIs: A specification-based approach. IEEE 22nd International Enterprise Distributed Object Computing Conference, 181-190. https://doi.org/10.1109/EDOC.2018.00031

Egele, M., Brumley, D., Fratantonio, Y., & Kruegel, C. (2013). An empirical study of cryptographic misuse in Android applications. Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, 73-84. https://doi.org/10.1145/2508859.2516693

Fahl, S., Harbach, M., Muders, T., Smith, M., Baumgärtner, L., & Freisleben, B. (2012). Why Eve and Mallory love Android: An analysis of Android SSL insecurity. Proceedings of the 2012 ACM Conference on Computer and Communications Security, 50-61. https://doi.org/10.1145/2382196.2382205

Gawer, A. (2014). Bridging differing perspectives on technological platforms: Toward an integrative framework. Research Policy, 43(7), 1239-1249. https://doi.org/10.1016/j.respol.2014.03.006

Goel, S., & Shawky, H. A. (2009). Estimating the market impact of security breach announcements on firm values. Information & Management, 46(7), 404-410. https://doi.org/10.1016/j.im.2009.06.005

Gordon, L. A., & Loeb, M. P. (2002). The economics of information security investment. ACM Transactions on Information and System Security, 5(4), 438-457. https://doi.org/10.1145/581271.581274

Gordon, L. A., Loeb, M. P., & Lucyshyn, W. (2003). Sharing information on computer systems security: An economic analysis. Journal of Accounting and Public Policy, 22(6), 461-485. https://doi.org/10.1016/S0167-4048(03)00003-X

Grover, V., Chiang, R. H. L., Liang, T. P., & Zhang, D. (2018). Creating strategic business value from big data analytics: A research framework. Journal of Management Information Systems, 35(2), 388-423. https://doi.org/10.1080/07421222.2018.1451951

Halfond, W. G. J., & Orso, A. (2005). AMNESIA: Analysis and monitoring for NEutralizing SQL-injection attacks. Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering, 174-183. https://doi.org/10.1145/1101908.1101935

Hasselbring, W., & Steinacker, G. (2017). Microservice architectures for scalability, agility and reliability in e-commerce.

Procedia Computer Science, 113, 243-246. https://doi.org/10.1016/j.procs.2017.08.154

Jovanovic, N., Kruegel, C., & Kirda, E. (2006). Pixy: A static analysis tool for detecting web application vulnerabilities. IEEE Symposium on Security and Privacy, 258-263. https://doi.org/10.1109/SP.2006.29

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. Science, 349(6245), 255-260. https://doi.org/10.1126/science.aaa8415

Kamiya, S., Kang, J. K., Kim, J., Milidonis, A., & Stulz, R. M. (2021). What is the impact of successful cyberattacks on target firms? Journal of Financial Economics, 139(3), 719-749. https://doi.org/10.1016/j.jfineco.2019.05.019

Kannan, K., Rees, J., & Sridhar, S. (2007). Market reactions to information security breach announcements: An empirical analysis. International Journal of Information Management, 27(2), 104-114. https://doi.org/10.1016/j.ijinfomgt.2006.12.001

Kou, G., & Lu, Y. (2025). FinTech: A literature review of emerging financial technologies and applications. Financial Innovation, 11(1), 1-34. https://doi.org/10.1186/s40854-024-00668-6

Kruchten, P., Nord, R. L., & Ozkaya, I. (2012). Technical debt: From metaphor to theory and practice. IEEE Software, 29(6), 18-21. https://doi.org/10.1109/MS.2012.167

Li, Z., Zou, D., Xu, S., Ou, X., Jin, H., Wang, S., Deng, Z., & Zhong, Y. (2018). VulDeePecker: A deep learning-based system for vulnerability detection. Proceedings of the 25th Annual Network and Distributed System Security Symposium. https://doi.org/10.14722/ndss.2018.23158

Livshits, B., & Lam, M. S. (2005). Finding security vulnerabilities in Java applications with static analysis. Proceedings of the 14th USENIX Security Symposium, 271-286. https://doi.org/10.1145/1065010.1065035

Lu, Y. (2019a). The blockchain: State-of-the-art and research challenges. Journal of Industrial Information Integration, 15, 80-90. https://doi.org/10.1016/j.jii.2019.04.002

Lu, Y. (2019b). Artificial intelligence: A survey on evolution, models, applications and future trends. Journal of Management Analytics, 6(1), 1-29. https://doi.org/10.1080/23270012.2019.1570365

Lu, Y. (2021). Technological innovation and the emergence of a new interdisciplinary field: Management Analytics. Nanotechnologies in Construction, 13(3), 181-192. https://doi.org/10.15828/2075-8545-2021-13-3-181-192

Lu, Y. (2022). Implementing blockchain in information systems: A review. Enterprise Information Systems, 16(12), 1876-1907. https://doi.org/10.1080/17517575.2021.2008513

Lu, Y. (2025). The current status and developing trends of Industry 4.0: A review. Information Systems Frontiers, 27(1), 215-234. https://doi.org/10.1007/s10796-021-10221-w

Lu, Y., & Xu, L. D. (2019). Internet of Things (IoT) cybersecurity research: A review of current research topics. IEEE Internet of Things Journal, 6(2), 2103-2115. https://doi.org/10.1109/JIOT.2018.2869847

Manès, V. J. M., Han, H., Han, C., Cha, S. K., Egele, M., Schwartz, E. J., & Woo, M. (2021). The art, science, and engineering of fuzzing: A survey. IEEE Transactions on Software Engineering, 47(11), 2312-2331. https://doi.org/10.1109/TSE.2019.2946563

Martin-Lopez, A., Segura, S., & Ruiz-Cortés, A. (2021). RESTest: Automated black-box testing of RESTful web APIs. Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis, 682-685. https://doi.org/10.1145/3460319.3469082

Mikalef, P., Boura, M., Lekakos, G., & Krogstie, J. (2020). The role of information governance in big data analytics driven innovation. Information & Management, 57(7), 103361. https://doi.org/10.1016/j.im.2020.103361

Nadi, S., Krüger, S., Mezini, M., & Bodden, E. (2016). Jumping through hoops: Why do Java developers struggle with cryptography APIs? Proceedings of the 38th International Conference on Software Engineering, 935-946. https://doi.org/10.1145/2884781.2884790

Nambisan, S., Lyytinen, K., Majchrzak, A., & Song, M. (2017). Digital innovation management: Reinventing innovation management research in a digital world. MIS Quarterly, 41(1), 223-238. https://doi.org/10.25300/MISQ/2017/41:1.03

Parker, G. G., & Van Alstyne, M. W. (2005). Two-sided network effects: A theory of information product design. Management Science, 51(10), 1494-1504. https://doi.org/10.1287/mnsc.1050.0400

Provost, F., & Fawcett, T. (2013). Data science and its relationship to big data and data-driven decision making. Big Data, 1(1), 51-59. https://doi.org/10.1089/big.2013.1508

Romanosky, S. (2016). Examining the costs and causes of cyber incidents. Journal of Cybersecurity, 2(2), 121-135. https://doi.org/10.1093/cybsec/tyw001

Russell, R., Kim, L., Hamilton, L., Lazovich, T., Harer, J., Ozdemir, O., Ellingwood, P., & McConley, M. (2018). Automated vulnerability detection in source code using deep representation learning. 17th IEEE International Conference on Machine Learning and Applications, 757-762. https://doi.org/10.1109/ICMLA.2018.00120

Scandariato, R., Walden, J., Hovsepyan, A., & Joosen, W. (2014). Predicting vulnerable software components via text mining. IEEE Transactions on Software Engineering, 40(10), 993-1006. https://doi.org/10.1109/TSE.2014.2340398

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J. F., & Dennison, D. (2015). Hidden technical debt in machine learning systems. Advances in Neural Information Processing Systems, 28, 2503-2511. https://doi.org/10.5555/2969442.2969519

Shin, Y., Meneely, A., Williams, L., & Osborne, J. A. (2011). Evaluating complexity, code churn, and developer activity metrics as indicators of software vulnerabilities. IEEE Transactions on Software Engineering, 37(6), 772-787. https://doi.org/10.1109/TSE.2010.81

Shmueli, G., & Koppius, O. R. (2011). Predictive analytics in information systems research. MIS Quarterly, 35(3), 553-572. https://doi.org/10.2307/23042796

Tanveer, F., Iradat, F., Iqbal, W., & Ahmad, A. (2025). Towards secure APIs: A survey on RESTful API vulnerability detection. Computers, Materials & Continua, 84(1), 1-30. https://doi.org/10.32604/cmc.2025.067536

Tiwana, A., Konsynski, B., & Bush, A. A. (2010). Platform evolution: Coevolution of platform architecture, governance, and environmental dynamics. Information Systems Research, 21(4), 675-687. https://doi.org/10.1287/isre.1100.0323

Wamba, S. F., Gunasekaran, A., Akter, S., Ren, S. J. F., Dubey, R., & Childe, S. J. (2017). Big data analytics and firm performance: Effects of dynamic capabilities. Journal of Business Research, 70, 356-365. https://doi.org/10.1016/j.jbusres.2016.08.009

Wassermann, G., & Su, Z. (2007). Sound and precise analysis of web applications for injection vulnerabilities. Proceedings of the 28th ACM SIGPLAN Conference on Programming Language Design and Implementation, 32-41. https://doi.org/10.1145/1250734.1250739

Wu, H. P., Liu, Z., Dong, H. Y., Lu, Y., & Xu, L. D. (2025). Revolutionizing internal auditing: Harnessing the power of blockchain. Enterprise Information Systems, 19(1-2), 2448003. https://doi.org/10.1080/17517575.2024.2448003

Xu, L. D., Lu, Y., & Li, L. (2021). Embedding blockchain technology into IoT for security: A survey. IEEE Internet of Things Journal, 8(13), 10452-10473. https://doi.org/10.1109/JIOT.2021.3060508

Xu, R., Zhu, J., Yang, L., Lu, Y., & Xu, L. D. (2024). Decentralized finance (DeFi): A paradigm shift in the FinTech. Enterprise Information Systems, 18(9), 2397630. https://doi.org/10.1080/17517575.2024.2397630

Yamaguchi, F., Golde, N., Arp, D., & Rieck, K. (2014). Modeling and discovering vulnerabilities with code property graphs. IEEE Symposium on Security and Privacy, 590-604. https://doi.org/10.1109/SP.2014.44

Zheng, X. R., & Lu, Y. (2022). Blockchain technology: Recent research and future trend. Enterprise Information Systems, 16(12), 1939895. https://doi.org/10.1080/17517575.2021.1939895