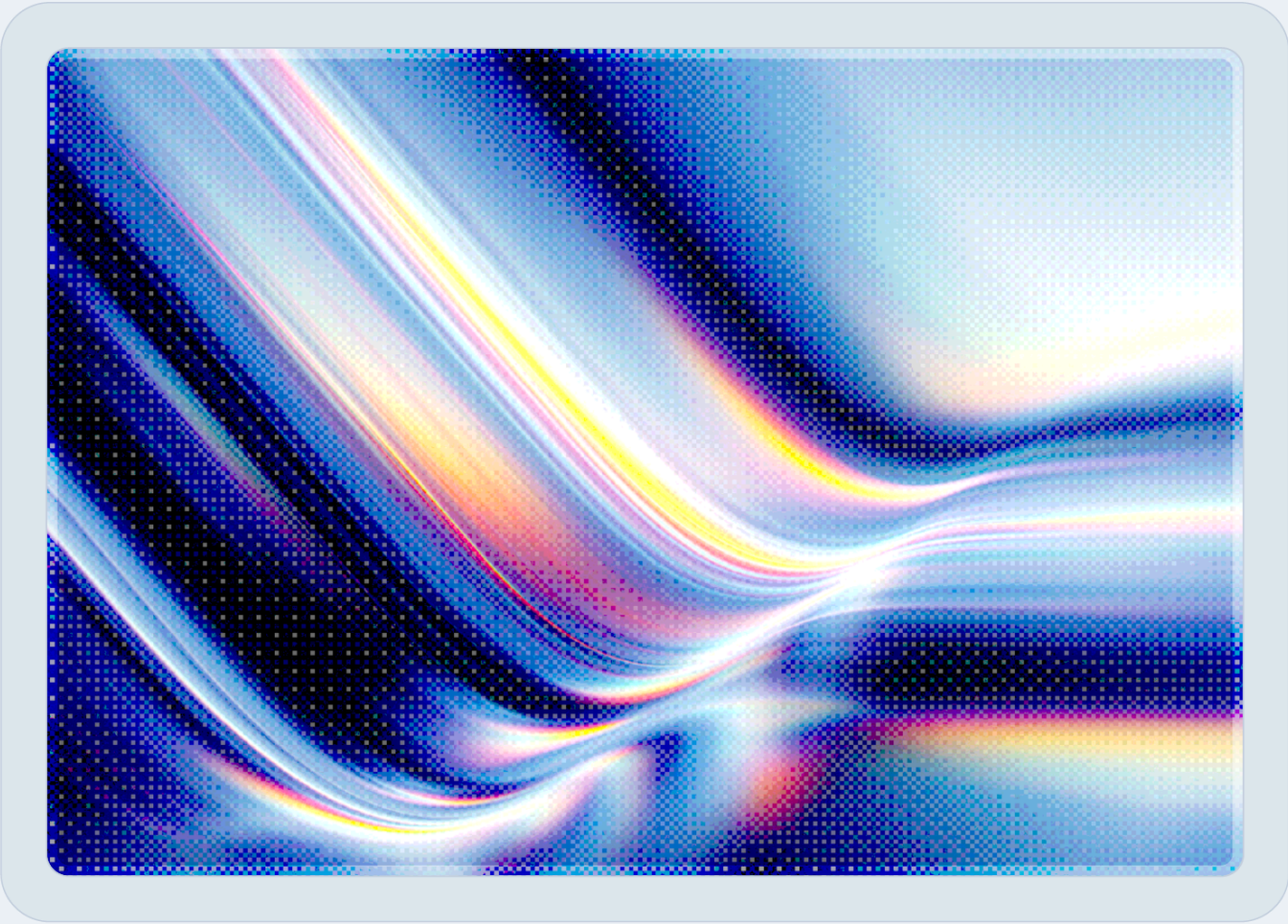
CSAI Foundation | Cloud Security Alliance

The Vibe Coding Governance Gap

Why AI Frameworks Overlook Citizen-Built Enterprise Apps

2026-06-02

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- Vibe coding—the practice of building functional software by directing AI models in natural language without reading or understanding the generated code—has spread from a fringe technique to broad enterprise adoption within roughly 16 months of Andrej Karpathy's coinage of the term.
- None of the major AI security frameworks in active use—NIST AI RMF, OWASP LLM Top 10, CSA MAESTRO, or CSA AICM—provide dedicated, accessible guidance for citizen developers who build and deploy AI-powered applications without professional security oversight.
- Empirical data from scanning thousands of production vibe-coded applications indicates pervasive missing security controls, with one study finding that zero out of 5,600 surveyed apps had CSRF protection, security headers, or properly scoped access policies [1].
- Shadow AI has evolved beyond tool adoption into shadow operations—employees building and deploying autonomous, data-processing applications on approved enterprise platforms without IT or security involvement, creating a category of risk that platform-level governance controls were not designed to catch.
- The industry needs a tiered, citizen-developer-accessible governance track within existing AI security frameworks; the current gap leaves a structurally ungovernated attack surface expanding at enterprise scale.

Background

On February 6, 2025, Andrej Karpathy—then recently departed from OpenAI—published a post on X that would reportedly be viewed more than 4.5 million times and eventually be named Word of the Year 2025 by Collins English Dictionary [2]. Karpathy described "a new kind of coding I call 'vibe coding', where you fully give in to the vibes, embrace exponentials, and forget that the code even exists." His described workflow was striking in its deliberate abandonment of traditional software craftsmanship: he dictated requirements via voice transcription, accepted all AI-generated changes without reading diffs, and handled errors by pasting them back to the AI for resolution [3]. The implicit message was that the quality of large language models had reached a threshold where comprehension of the generated code

was no longer a prerequisite for producing working software. In a February 2026 retrospective, Karpathy himself acknowledged the outsized impact of what he called "a shower of thoughts throwaway tweet" [4].

The phenomenon Karpathy named did not originate with his post—it codified a pattern that had already been emerging across enterprise environments. The low-code and no-code movement had spent years lowering the technical barrier to application creation; AI-assisted generation dramatically reduced it further, enabling business users without coding backgrounds to build functional applications. By 2025, Gartner had projected that 70% of new enterprise applications would be built using low-code or no-code technologies, shifting the locus of software creation from professional developers to business analysts, operations staff, and domain experts who may have no formal engineering background [5]. IDC research from the same period found that 56% of employees use unauthorized AI tools at work and only 23% use IT-governed alternatives [6]. The result is a category of enterprise software development that sits almost entirely outside established security and governance structures.

The risk profile introduced by vibe coding is structurally distinct from the risks associated with professional developers using AI coding assistants. When a trained engineer uses GitHub Copilot or similar tooling, they typically review generated code, apply professional judgment about correctness and security, and integrate the output into an established development and review workflow. The citizen developer using Replit, Lovable, Cursor, or similar prompt-to-app platforms neither possesses the background to perform that review nor, in the paradigm Karpathy described, intends to—meaning code can reach production without having been understood by any human who holds accountability for its security properties. The resulting applications may be functionally adequate while lacking basic security controls—and the organizations that depend on them may not know they exist until a breach occurs.

Security Analysis

The Framework Gap

The major AI security frameworks published and updated since 2024 represent genuine advances in security thinking, but they share a common scope assumption: they address technically sophisticated actors—developers, security architects, DevSecOps teams, AI red teamers—operating within formal organizational structures with defined governance processes. This assumption is so deeply embedded that none of the frameworks acknowledge citizen developers as a distinct actor class requiring dedicated guidance.

NIST AI RMF 1.0 and its July 2024 generative AI supplement, NIST AI 600-1, organize risk management across four functions—Govern, Map, Measure, and Manage—and identify 12 generative AI-specific risk categories [7]. The framework's actors are organizational entities with documented roles. A business analyst who built a client data processing tool on Microsoft Copilot Studio over a weekend has no role defined in the NIST model and no pathway to apply its controls. The OWASP LLM Top 10 for 2025 and the companion OWASP GenAI Solutions Reference Guide are similarly scoped: they explicitly address developers, AppSec teams, CISOs, and data scientists [8]. OWASP does maintain a separate Citizen Development Top 10 Security Risks project and a Low-Code/No-Code Top 10, but these predate the vibe-coding era and have not been substantively updated to address AI-generated application creation [9]. CSA's MAESTRO framework—a seven-layer agentic AI threat modeling tool—is designed for security practitioners performing structured threat modeling, and its implicit audience is teams operating organized enterprise or security-program actors at each layer [10]. The CSA AI Controls Matrix, released July 2025, maps 243 control objectives across 18 domains to five actor roles, but the citizen developer falls ambiguously between "Application Provider" and "AI Customer"—both of which carry control obligations that a typical citizen developer is structurally unequipped to fulfill [11].

The gap is not a criticism of these frameworks' quality; it reflects the speed at which vibe coding scaled relative to the policy development cycle. IBM's 2025 Cost of a Data Breach report found that 63% of the 600 organizations it surveyed had no AI governance policies in place [12]—a baseline that reflects how far adoption has outpaced governance infrastructure. The frameworks arrived before the problem they needed to address was fully visible.

The Provenance Problem

Professional software development produces artifacts—version-controlled source code, dependency manifests, software bills of materials, commit histories—that give security teams the raw material for vulnerability assessment, license compliance, and incident investigation. Vibe-coded applications routinely produce none of these. An application created via a prompt-to-app platform may exist only as a deployed service with no retrievable source code, no dependency inventory, and no record of the AI model version or prompt that generated it. When a vulnerability is discovered, there is no code to patch in any conventional sense; the remediation path is to re-prompt the AI, generating a new application whose security properties are as uncertain as the original's.

This provenance gap has measurable consequences at the CVE level. Georgia Tech's Vibe Security Radar, launched May 2025 to track vulnerabilities formally attributable to AI-generated code, documented a sharply accelerating trend: 6 CVEs in January 2026, 15 in February, and 35 in March 2026 alone—more than all of 2025 combined [13]. The trajectory may reflect expanding production deployment of AI-generated code, increasing researcher attention to the newly named category, or both

—distinguishing between these drivers from CVE counts alone is methodologically difficult, though the growth of platform deployment data points to production expansion as a material contributor. The vulnerability classes being documented are not novel or exotic. An October 2025 scan of 5,600 vibe-coded production applications found that zero had CSRF protection, zero had standard security headers including CSP, HSTS, or X-Frame-Options, and every application introduced server-side request forgery vulnerabilities in URL-handling features [1]. These are foundational web security controls whose absence reflects not sophisticated evasion but simple unawareness on the part of both the AI models and the citizen developers directing them.

Credential exposure compounds the provenance problem. GitGuardian documented 28.65 million hardcoded secrets in public GitHub commits in 2025, a 34% year-over-year increase, with commits co-authored by Claude Code exposing secrets at more than twice the rate of human-written code—3.2% versus 1.5% [14]. AI service keys specifically grew to 1,275,105 leaked instances in 2025, an 81% year-over-year increase. When a citizen developer's application has no audit trail and no code review history, there is no mechanism to detect that it contains a hardcoded database credential or API token until that credential is discovered or abused.

Shadow IT Amplified

The governance challenge of shadow IT—employees using unsanctioned technology services—has occupied enterprise security programs for more than a decade. Vibe coding does not simply extend that problem; it changes its character in ways that existing playbooks are poorly positioned to address. Traditional shadow IT involved employees adopting approved-category services, such as file sharing or messaging, through unapproved vendors. The standard response—discover via network monitoring, block via policy, redirect to sanctioned alternatives—was operationally tractable because the tools and the risks were familiar.

Vibe-coded shadow applications are different on several dimensions. First, they are often built on sanctioned enterprise platforms. Microsoft Power Platform, ServiceNow, Salesforce, and similar environments are explicitly approved for business use; the governance question is whether the specific application a business unit analyst built on Power Platform last week has been reviewed for security. The platform is sanctioned; the application is not. Second, these applications increasingly incorporate AI API calls, creating a category that analysts have termed "shadow operations"—autonomous agents built by non-technical employees on foundation model APIs that process business data, send external communications, and make decisions with no IT visibility [15]. Third, the economics of detection and response have shifted: blocking a category of tool is feasible, but auditing every application produced by a workforce of citizen developers on platforms the organization itself has approved is a fundamentally different operational challenge.

IBM's 2025 Cost of a Data Breach report found that data breaches involving shadow AI cost an average of $670,000 more than other incidents, pushing the average shadow AI breach to $4.63 million [12]. A separate analysis found that 20% of organizations had already experienced breaches linked to shadow AI by 2025, and 49% expected a shadow AI-related incident within the following 12 months [6]. The Moltbook incident illustrates the mechanism concretely: a vibe-coded social network launched with no hand-written code was found, within three days of launch, to have a Supabase API key exposed in client-side JavaScript with Row Level Security completely disabled, exposing 1.5 million API authentication tokens and 35,000 email addresses [16]. The Cal AI breach followed a similar pattern: 14.59 GB of data containing approximately 3.2 million user records was exposed via an unauthenticated Google Firebase backend in a vibe-coded consumer application [17]. A security researcher reported in April 2026 that Lovable projects created before November 2025 were accessible to other users, with one account able to view another user's source code, database credentials, and customer data; Lovable disputed the severity of the characterization [18].

The Compliance Blindspot

Regulatory frameworks for data privacy and security—GDPR, HIPAA, SOC 2, PCI DSS—impose obligations on organizations based on the categories of data their applications process, not on how those applications were built. A vibe-coded application that handles protected health information carries the same HIPAA exposure as one written by a team of professional developers following a formal SDLC. The difference is that the professional development team will typically have security review, penetration testing, and audit logging in place; the citizen developer almost certainly will not. CVE-2025-48757 documented this risk concretely: Lovable-generated Supabase projects were found to have insufficient Row-Level Security policies—the AI generated the database layer but not the corresponding access controls—exposing data across more than 170 production applications [24].

The compliance exposure is compounded by the absence of data processing transparency. GDPR's accountability principle requires that organizations be able to demonstrate how personal data flows through their systems. An organization that cannot identify all the applications processing personal data —because those applications were built by business users without IT notification—cannot meet this requirement. EU AI Act full enforcement for high-risk systems begins August 2, 2026, adding another regulatory layer that citizen-built AI applications may inadvertently trigger while operating beneath the organization's governance visibility [19]. Governance programs face a structural barrier on this problem: risk committees cannot verify security controls in applications they cannot enumerate, and legal teams cannot sign off on data processing activities they are unaware of.

Agentic Capability Without Agentic Controls

The final dimension of the governance gap concerns the capabilities that vibe-coded applications increasingly incorporate. Citizen developers building on modern AI-assisted platforms are not merely creating static CRUD applications; they are building applications that call AI APIs, chain model invocations, and in some cases deploy autonomous agents that act on external systems. These capabilities are the same ones that security frameworks like MAESTRO and the forthcoming NIST Agentic Profile are designed to govern—but those frameworks assume a security-aware practitioner at the design stage. A business analyst building an expense approval agent on a no-code platform is unlikely to reason about prompt injection attack surfaces, excessive agency, or the cascading effects of an agent that can send emails and modify records.

The OWASP LLM Top 10 category LLM06 (Excessive Agency) describes exactly the risk this creates: agents that fail to wire authentication middleware into subsequent components, where the vulnerability lives in the integration between components rather than in any individual file [8]. When that integration is generated by an AI responding to a natural language prompt, with no human reviewing the output for security implications, the likelihood that such controls will be correctly implemented is low. Apiiro research across Fortune 50 enterprises found 322% more privilege escalation paths in AI-generated code compared to human-written code, and 153% more design flaws [20]. These are not gaps that automated scanning tools reliably catch because they reflect architectural decisions—or the absence of them—rather than discrete code-level defects.

Recommendations

Immediate Actions

Organizations should begin by establishing visibility before attempting control. IT and security teams that do not currently inventory citizen-built applications have no baseline against which to assess risk or detect new exposure. A practical first step is to audit the AI features within already-sanctioned platforms —Microsoft Power Platform, Salesforce, ServiceNow, Google Workspace—for applications that handle sensitive data categories. Most major platforms now offer governance dashboards that enumerate applications, their owners, and their data connections; these dashboards are frequently underutilized.

Security teams should also implement data loss prevention policies at the API gateway layer to detect outbound connections from citizen-built applications to external AI services. Applications that route business data to third-party LLM providers outside sanctioned channels represent a direct data exfiltration risk regardless of whether the data transfer is intentional. Alongside this, organizations should

establish a lightweight registration process—deliberately not a heavyweight review process at this stage—that allows citizen developers to declare new applications. The goal of this initial step is awareness, not gatekeeping; many citizen developers are likely to comply with a frictionless registration requirement when it is paired with non-punitive framing, though monitoring for unregistered applications should accompany any registration program to catch those who do not.

Short-Term Mitigations (30–90 Days)

Within a quarter, organizations should develop a tiered governance model that distinguishes between internal tools handling non-sensitive data, applications processing personally identifiable information or regulated data, and applications that incorporate AI API calls or agentic functionality. These three tiers warrant different levels of security review, not a single heavyweight gate that will be circumvented in practice. Low-code governance frameworks, including the Center of Excellence model recommended by practitioners such as those at Superblocks and Zenity—both commercial vendors in the low-code governance space—provide workable reference architectures for this kind of tiered intake process [21] [22].

Organizations should extend their software composition analysis and secrets-scanning tooling to cover outputs from no-code and low-code platforms where export is supported. Some platforms allow source code export; where they do, automated scanning should be mandatory before an application is deployed to handle regulated data. Where source code export is not available, organizations should establish contractual requirements with platform vendors for security attestations and audit logging capabilities.

Security awareness programs should be updated to explicitly address vibe coding risks. Developers and non-technical staff alike frequently lack awareness that AI-generated code may embed credentials, omit authentication controls, or create data exposure through misconfigured backends. Training should be concrete, scenario-based, and tied to the specific platforms employees are using, rather than generic AI security guidance.

Strategic Considerations

At the industry level, the most important long-horizon change is the development of a citizen-developer-accessible governance track within existing AI security frameworks. This track should not be a separate framework—framework proliferation is itself a governance problem—but rather a simplified, tiered extension of existing work. CSA's AICM is a natural candidate for this extension: its actor role taxonomy could be augmented with a "Citizen Developer / Non-Technical Deployer" role that carries a

reduced but specific set of control obligations appropriate to the actor's capabilities and context. Similarly, OWASP's Low-Code/No-Code Top 10 project deserves investment to update its scope to the prompt-to-app and vibe-coding context it was not designed to address.

Platform vendors bear responsibility that neither enterprise policy nor security frameworks can fully substitute for. The technical controls available through sanctioned platforms—mandatory security policy templates, automatic secrets scanning at deploy time, required authentication configuration before an application can process external data—represent a defense-in-depth layer that operates without requiring citizen developers to possess security knowledge they do not have. Governance requirements for platform vendors should evolve to mandate these baseline controls rather than treating them as optional features. The Power Platform March 2026 governance update, which introduced enhanced AI governance capabilities including expanded policy enforcement controls, indicates that this direction is technically feasible [23].

Finally, the regulatory frameworks beginning enforcement in 2026—including the EU AI Act—will need implementation guidance that acknowledges the citizen developer as a distinct deployment pathway. Organizations cannot be expected to satisfy high-risk system requirements for applications they did not know existed, but they can be expected to implement discovery and oversight processes that make ungoverned deployment the exception rather than the norm. Developing that guidance in collaboration with regulators, before enforcement creates perverse incentives for concealment, would serve both organizations and the public interest.

CSA Resource Alignment

Several CSA resources bear directly on the governance gap described in this note, though none were designed with citizen development as a primary use case.

The **CSA AI Controls Matrix** (https://cloudsecurityalliance.org/artifacts/ai-controls-matrix) is the most directly applicable framework for organizations seeking to map control obligations. Its 243 control objectives cover the domains most relevant to vibe-coded app exposure—credential management, access control, data privacy, audit logging—and its actor role taxonomy is the natural starting point for developing a citizen developer extension. The gap is the absence of that extension; the AICM's current roles assume actors with organizational accountability structures that typical citizen developers do not operate within.

CSA MAESTRO (https://labs.cloudsecurityalliance.org/maestro/) provides the threat modeling vocabulary for the agentic risk dimension of this problem. Its seven-layer model—and particularly Layer 3 (Agent Frameworks) and Layer 7 (Agent Ecosystem)—maps to the attack surfaces introduced when

citizen-built applications incorporate AI API calls or autonomous agent behavior. Security architects using MAESTRO to assess enterprise AI risk should explicitly scope their threat models to include applications built outside formal development pipelines, as these represent a material and growing portion of the enterprise AI surface.

The **CSA AI Organizational Responsibilities** series (https://cloudsecurityalliance.org/artifacts/ai-organizational-responsibilities-core-security-responsibilities) addresses the organizational accountability question that sits at the heart of the governance gap. Its RACI model for AI deployment responsibilities is a workable starting point for assigning ownership of citizen-built applications within an enterprise security program, though it will require adaptation to address the scenario where no organized team holds clear accountability for a given application.

The **CSA Secure Vibe Coding Guide** (https://cloudsecurityalliance.org/blog/2025/04/09/secure-vibe-coding-guide), published April 2025, represents CSA's most direct current response to this problem. It addresses AI-assisted code generation by non-developers and the security risks of LLM-generated code entering production. Organizations looking for immediately actionable guidance specific to vibe-coded app risks should treat this guide as the starting point while the broader framework community develops more comprehensive governance coverage.

The **CSA STAR for AI** program (https://cloudsecurityalliance.org/star/ai) offers a market-based mechanism for platform vendors to demonstrate security assurances—including the baseline controls for no-code and low-code platforms that would most directly reduce citizen developer risk at the point of creation. Organizations selecting citizen development platforms should consider STAR for AI certification as a procurement criterion.

References

[1] Escape.tech. "5,000 vibe-coded apps just proved shadow AI is the new S3 bucket crisis." VentureBeat, October 2025.

[2] Collins English Dictionary. "Vibe coding – Word of the Year 2025." Via Wikipedia: Vibe Coding, 2025.

[3] Karpathy, Andrej. "Original vibe coding post." X (formerly Twitter), February 6, 2025.

[4] Karpathy, Andrej. "One-year retrospective post." X (formerly Twitter), February 2026.

[5] Gartner. "How to Effectively Govern Low-Code Platforms Across Your Organization." Gartner, 2025.

[6] Vectra AI. "Shadow AI explained: risks, costs, and enterprise governance." Vectra AI, 2025. Citing IDC 2025 and IBM 2025 data.

[7] NIST. "NIST AI 600-1: Artificial Intelligence Risk Management Framework – Generative Artificial Intelligence Profile." National Institute of Standards and Technology, July 2024.

[8] OWASP. "OWASP Top 10 for LLM Applications 2025." OWASP GenAI Security Project, 2025.

[9] OWASP. "OWASP Top 10 Low-Code/No-Code Security Risks." OWASP, 2023.

[10] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." CSA Blog, February 2025.

[11] Cloud Security Alliance. "AI Controls Matrix." CSA Artifact, July 2025.

[12] IBM Security. "Cost of a Data Breach Report 2025." IBM Security, 2025.

[13] Cloud Security Alliance Labs. "Vibe Coding's Security Debt: The AI-Generated CVE Surge." CSA Labs, 2026. Citing Georgia Tech Vibe Security Radar data.

[14] GitGuardian. "State of Secrets Sprawl 2026." GitGuardian, 2026.

[15] CIO. "Shadow AI morphs into shadow operations." CIO Magazine, 2026.

[16] Wiz Security. "Exposed: Moltbook Database Reveals Millions of API Keys." Wiz Security Blog, 2026.

[17] Kiteworks. "Cal AI Data Breach: 3 Million Users' Health Data Exposed." Kiteworks, March 2026.

[18] The Register. "Lovable denies data leak." The Register, April 2026.

[19] SecurePrivacy. "AI Risk & Compliance 2026: Enterprise Governance Overview." SecurePrivacy, 2026.

[20] Apiiro. "4x Velocity, 10x Vulnerabilities: AI Coding Assistants Are Shipping More Risks." Apiiro Blog, 2025.

[21] Superblocks. "6-Step Framework for Citizen Developer Governance in 2026." Superblocks Blog, 2026.

[22] Zenity. "Security Governance Framework for Low-Code/No-Code Development." Zenity White Paper, 2025.

[23] Windows News AI. "Power Platform March 2026 Update: Agentic Apps, Enhanced Governance and AI Development Acceleration." WindowsNews.ai, March 2026.

[24] Palmer, Matt. "CVE-2025-48757: Lovable Supabase Row Level Security Breakdown." MattPalmer.io, May 2025.