OWASP citizen development top 10

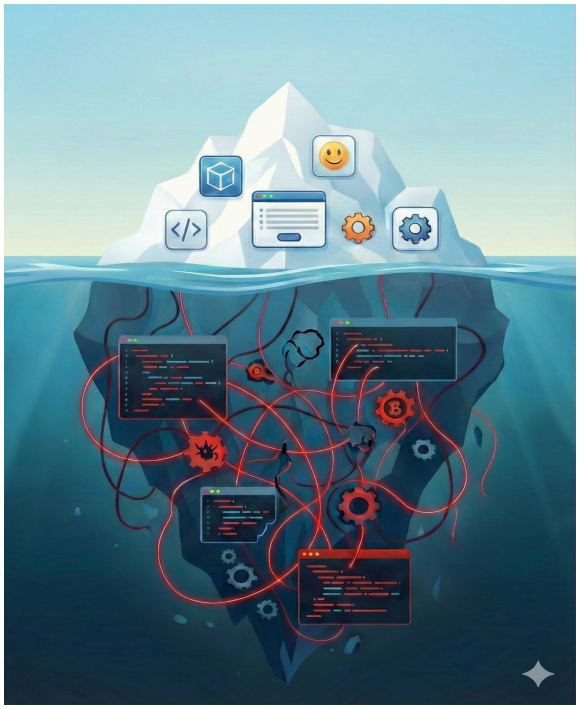
CD-SEC-01

Blind Trust

Blind trust refers to the uncritical acceptance of outputs, recommendations, or artifacts produced by an automated system, platform, or model without independent verification of their accuracy, integrity, or security. In software development contexts—particularly AI-assisted and low-code/no-code environments—blind trust manifests when developers assume that system-generated code, templates, or components are inherently safe or correct, leading to overlooked vulnerabilities and systemic security risks.

Blind Trust in AI-assisted and low-code/no-code development should be treated as a fundamental operating assumption by security teams because it defines a critical risk profile. Citizen developers often lack security training and experience several cognitive biases during their development process.

- **Automation Bias:** Citizen developers falsely assume that platform-generated code and pre-built components are inherently secure simply because the platform provides them.
- **Availability Heuristic:** Developers prioritize easily accessible code and components over optimal, secure alternatives.
- **Anchoring Bias:** Developers fixate on the initial templates or designs presented to them, leading to the widespread adoption of insecure defaults and vulnerable templates.



The Faces of Blind Trust: Recognizing Our Own Biases

"Trusting Tina" and the Automation Trap:

Tina falls for **Automation Bias**, assuming the AI is inherently correct. When the tool hallucinates and generates a script with dangerous administrative access instead of the restricted permissions she requested, she deploys it without question. Her blind trust in the machine exposes the entire cloud environment. To prevent this, require reviews for sensitive outputs like administrative access before implementation.

"Speedy Steve" and the Availability Trap:

Rushing to meet a deadline, Steve grabs the first login template he sees on a public marketplace. Falling for the **Availability Heuristic**, he assumes that because the code is easily accessible and looks professional, it must be safe. Instead, he unknowingly imports a hidden crypto-miner that hijacks the company's backend. This highlights why convenience is not security—developers must always use vetted code and components.

"Baseline Ben" and the Anchoring Trap:

Ben generates an initial code block that contains a vulnerability. Due to **Anchoring Bias**, he locks onto this first output as the "correct" foundation. Instead of questioning it, he warps the rest of his application to fit this insecure structure and repeatedly reuses the flawed code. To prevent this, teams must ensure the "anchor" is solid by reviewing initial code before it dictates the rest of the build.

Prevention – Blind Trust Secure Framework

Enforce Secure Defaults

This prevention strategy addresses the fundamental behavioral biases of Citizen Developers towards simplicity, speed, and immediate access. This aims to recognize that developers often choose the path of least resistance, and design workflows that make the secure option the easiest and most convenient choice.

- Design platforms with secure-by-default configurations, ensuring that default settings are inherently safe and difficult to disable.
- Establish a standardized and vetted component library so developers can safely reuse trusted code and templates.

Extend the Scope of Governance

This strategy addresses the disconnect between our classifications of citizen developed apps. We extend the scope of security by treating citizen development apps as governed extensions of our official environment, moving beyond the core platform's API to ensure security controls and continuous, real-time oversight extend into the configuration and data access of every single application.

- Integrate these controls into a governance framework that enforces compliance and ensures consistent application of security policies across all AI-assisted and low-code/no-code projects.

Secure Innovation Through Trust

We secure the innovation through trust at every stage by shifting security into an integrated component of application quality throughout the entire development lifecycle.

- Provide just-in-time security guidance embedded directly within the development environment to reinforce secure decision-making during coding and configuration.
- Offer structured training resources and reference guides to improve citizen developers' understanding of common vulnerabilities and secure development practices.
- Implement a pre-deployment security auditing and validation system to automatically review all generated or modified code before release.