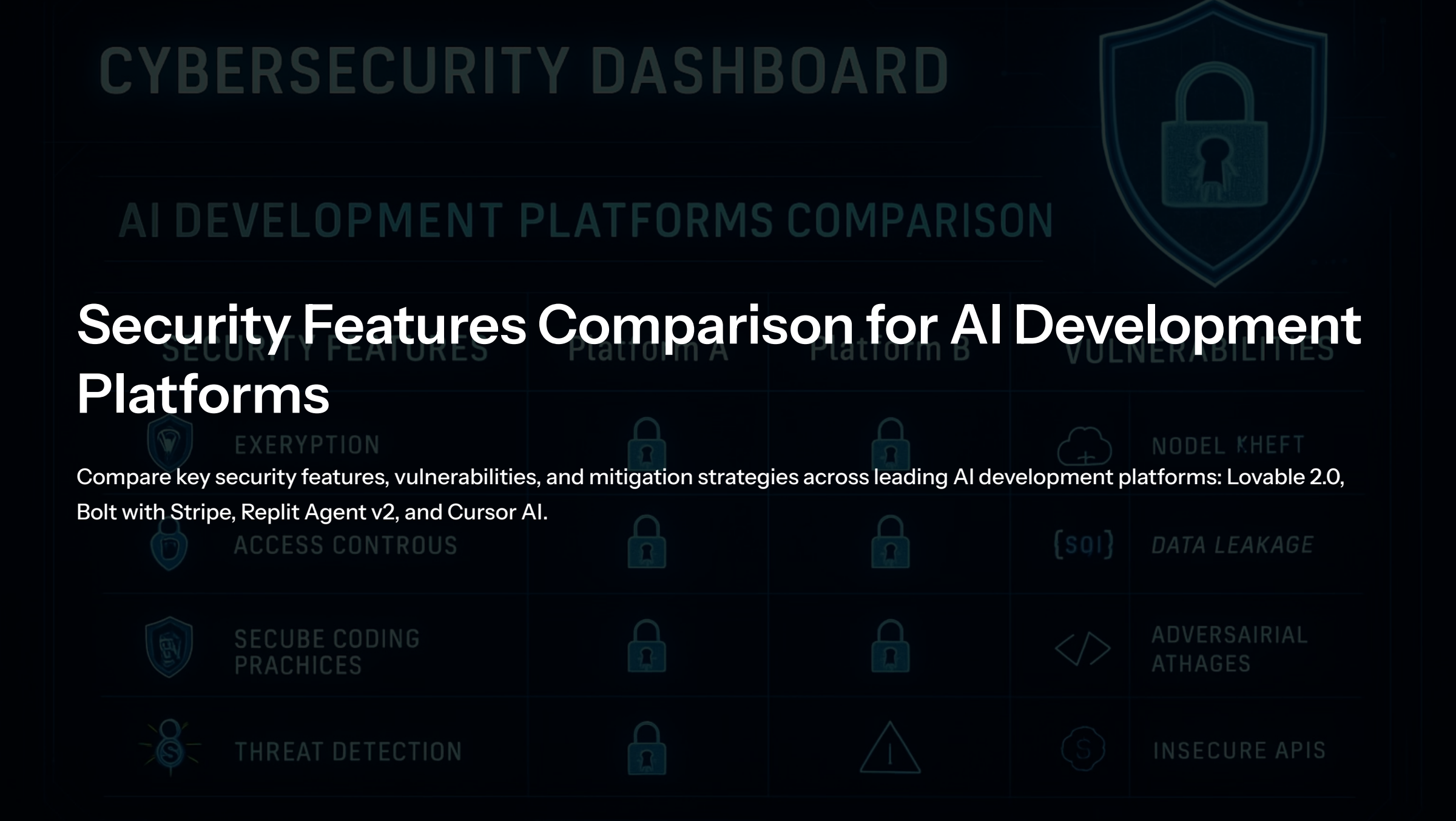


Security Features Comparison for AI Development Platforms

Compare key security features, vulnerabilities, and mitigation strategies across leading AI development platforms: Lovable 2.0, Bolt with Stripe, Replit Agent v2, and Cursor AI.

Platform Key Security Features

Lovable 2.0

- End-to-End encryption for prompts & sessions
- Clear source-code ownership
- Flexible self-host / code deployment control
- Data-privacy controls
- Security Scan for Supabase apps

Bolt (with Stripe)

- "Pit-of-Success" design encourages safe defaults
- Built-in secrets manager (encrypted env vars)
- Automatic ORM guards against SQLi
- GCP-hosted infra with DDoS protection & HTTPS
- Version control integration
- SOC 2 Type II & regular pen-testing

Replit Agent v2

- SOC 2 Type II controls
- Regular pen-testing
- VS Code base inherits upstream patches
- "Privacy Mode" (limits data persistence)
- Zero-retention claims with AI providers
- Secrets management via VS Code extensions
- Public vulnerability-disclosure program

Cursor AI

- VS Code foundation - inherits rapid security patches
- Privacy Mode to limit data sharing
- Zero-data-retention commitments with model providers
- Secrets management extensions
- Vulnerability-disclosure program

Common Vulnerabilities

Lovable 2.0

AI-generated code flaws (XSS, auth bypass, etc.) if not reviewed

Exposed secrets when hard-coded by AI / user

Misconfigured access control in generated logic

Insecure or outdated dependencies

Bolt (with Stripe)

Insecure API-key management (keys leaked to frontend)

Unsafe Stripe-webhook handling (no sig verification)

AI-generated code flaws (XSS, CSRF, payment logic)

Supabase RLS misconfigurations

Vulnerable third-party libraries

Replit Agent v2

AI-generated code flaws despite safeguards

Improper use of secrets when not using manager

Complex logic mistakes in generated app

Outdated / vulnerable dependencies

Mis-configured CORS or other env issues

Resource-exhaustion / leaks

Cursor AI

"Rules-file backdoor" risk (malicious hidden instructions)

AI-generated code flaws (XSS, SQLi, etc.)

Vulnerable dependencies

Potential VS Code-level CVEs

Data leakage when not in privacy mode

Insecure API-key handling

Mitigation Strategies

Lovable 2.0

- Mandatory code review of all AI output
- Use Git-based version control & PRs
- Add custom security layers (WAF, rate-limits, etc.)
- Harden deployment (HTTPS, env separation)
- Run Lovable's security scan & dependency scanners
- Apply secure-prompt-engineering patterns

Bolt (with Stripe)

- Store keys in env vars; follow Stripe key-rotation best practices
- Verify webhook signatures & process asynchronously
- Code-review payment flows
- Lock-down Supabase RLS policies
- Continuous dependency scanning
- Apply secure-prompt techniques to reduce harmful code

Replit Agent v2

- Leverage built-in secrets, ORM, Git workflows
- Enforce mandatory PR-based code review
- Secure-prompt engineering & server-side validation
- Automated dependency and SBOM scans
- Unit / integration testing & log monitoring
- Follow Replit security best practices

Cursor AI

- Scrutinise & version-control .cursor/rules files as code
- Critical code review for all AI output
- Keep Cursor updated; apply VS Code patches
- Enable Privacy Mode in production projects
- Secure-prompt engineering & secrets in env vars
- Dependency scans for VS Code extensions
- Automated tests (unit, integration, security)

AI Safety Tool Comparison

Tool	What It Already Does for Safety	What Could Still Go Wrong	What You Should Do
Lovable 2.0	Runs a quick security check on your app and locks down data in transit.	The AI might write sloppy code or leave secret keys visible.	Read the code before shipping, hide keys in environment files, and rerun the built-in scan.
Bolt + Stripe	Stores secrets safely, uses a safer database layer, and follows strict audits (SOC 2).	Stripe keys or webhooks might be set up the wrong way, or older libraries could be risky.	Keep keys out of the front end, double-check webhooks, and update your packages.
Replit Agent v2	Has privacy mode, keeps secrets in one place, and is independently security-tested.	AI-written code could have bugs; old packages might slip in.	Turn on privacy mode, insist on a pull-request review, and let automated tools patch dependencies.
Cursor AI	Offers a "privacy mode" and insists on clear rule files that guide the AI.	Sneaky rules or AI mistakes could introduce security holes.	Treat rule files like regular code—review them, keep privacy mode on, and test changes before going live.

Prompting Aspects Comparison

Lovable 2.0

Primary Focus: Visual Design & UI/UX

Level of Abstraction: Describe visual outcome & user flow

Handling Ambiguity: Use chat agent to discuss & plan options

Bolt

Primary Focus: Full-Stack Logic & Payment Flows

Level of Abstraction: Specify frontend/backend logic & data models

Handling Ambiguity: Requires specific implementation details

Replit Agent v2

Primary Focus: High-Level Objectives & End Result

Level of Abstraction: Define goals, allow agent implementation

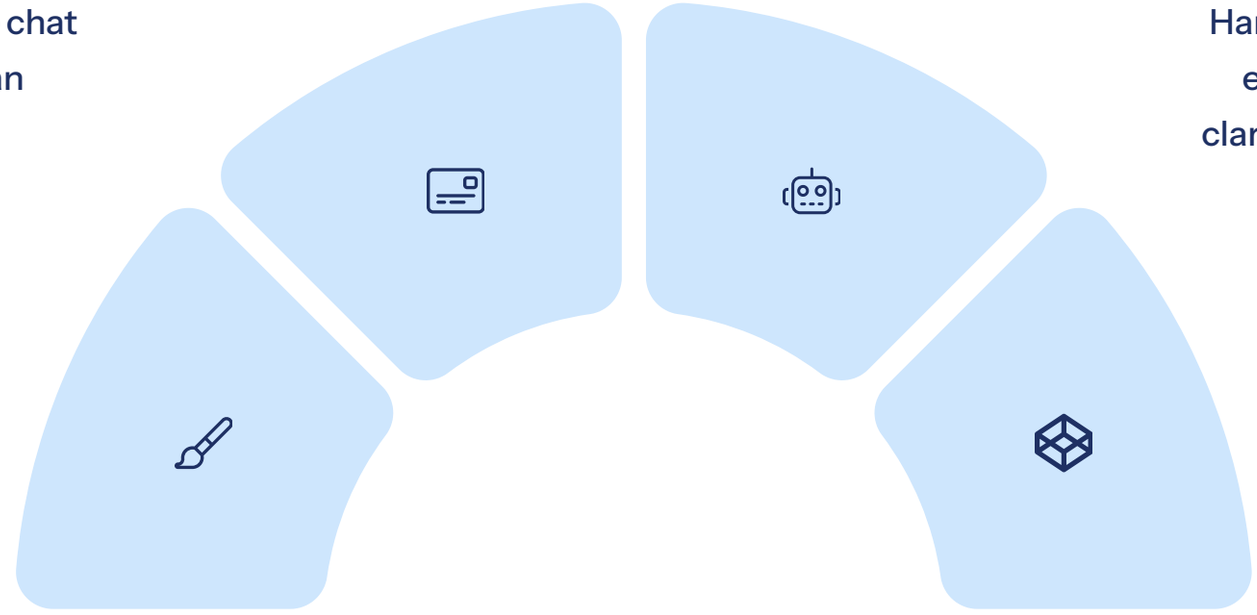
Handling Ambiguity: Agent forms hypotheses & asks for guidance

Cursor AI

Primary Focus: Code Structure, Logic & Technical Details

Level of Abstraction: Detail specific algorithms, libraries & file paths

Handling Ambiguity: Agent explores code, requires clarification or model choice



Key Feature Leverage & Feedback Mechanisms

Lovable 2.0

Key Feature Leverage: Prompt visual changes, use multiplayer context

Feedback Mechanism: Feedback on visual editor output & chat

Resource Management Emphasis: Combine related visual/functional requests (Credits)

Ideal Prompt Style: Visually descriptive, collaborative

Bolt

Key Feature Leverage: Prompt Stripe/Figma/Expo specifics, target files

Feedback Mechanism: Feedback via code diffs & specific file changes

Resource Management Emphasis: Target files, use diffs, lock files (Tokens)

Ideal Prompt Style: Structured, imperative, full-stack context

Replit Agent v2

Key Feature Leverage: Prompt based on real-time preview, guide autonomy

Feedback Mechanism: Feedback based on real-time preview & agent progress

Resource Management Emphasis: Comprehensive prompts, strategic feedback (Checkpoints)

Ideal Prompt Style: Objective-oriented, iterative feedback

Cursor AI

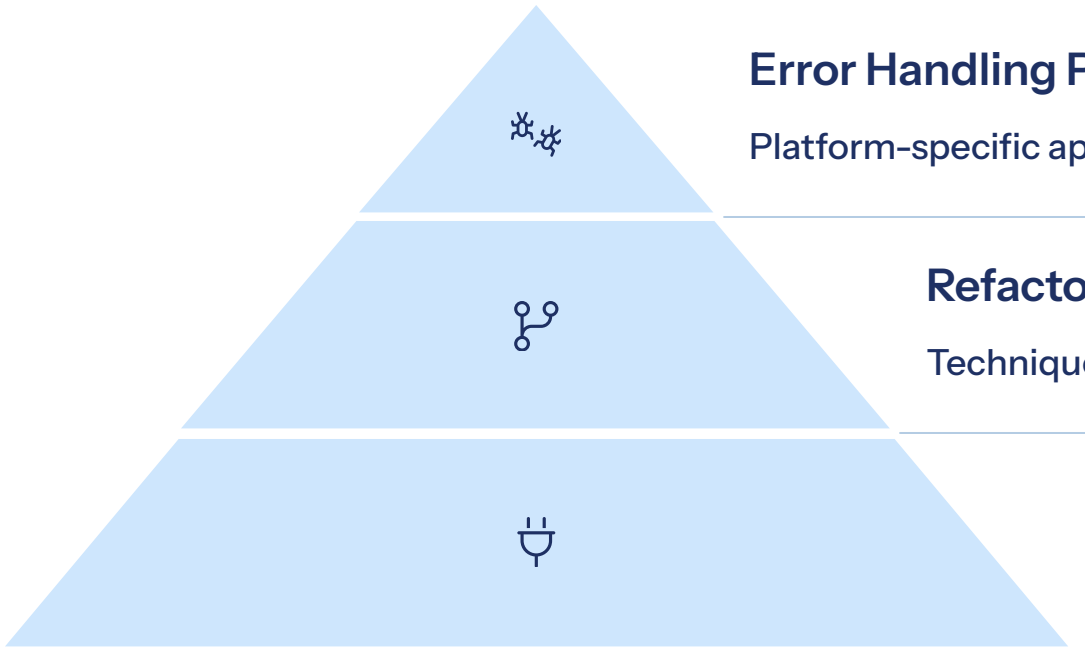
Key Feature Leverage: Prompt web search, terminal commands, rule generation, model selection

Feedback Mechanism: Feedback on code changes, exploration results, agent chat

Resource Management Emphasis: Select cost-effective models, targeted prompts (Tokens)

Ideal Prompt Style: Technically specific, exploratory, context-aware

Specialized Prompting Techniques



Error Handling Prompts

Platform-specific approaches to debugging and fixing issues

Refactoring Prompts

Techniques for improving existing code across platforms

Integration Prompts

Methods for connecting systems and APIs

Error Handling Prompts:

Lovable 2.0: Describe visual/functional error, ask agent to fix

Bolt: Specify error scenario, expected behavior, relevant code sections

Replit Agent v2: Describe error observed in preview, let agent hypothesize fix

Cursor AI: Provide error logs, ask agent to explore/debug specific files

Refactoring Prompts:

Lovable 2.0: Describe desired visual/UX change

Bolt: Specify code sections, desired pattern, performance goals

Replit Agent v2: Define refactoring goal, let agent propose approach

Cursor AI: Specify files/functions, refactoring strategy, technical constraints

Integration Prompts:

Lovable 2.0: Describe desired integration visually/functionally

Bolt: Specify API endpoints, data mapping, payment logic

Replit Agent v2: Define integration goal, let agent handle implementation

Cursor AI: Specify libraries, API docs (via web search), error handling