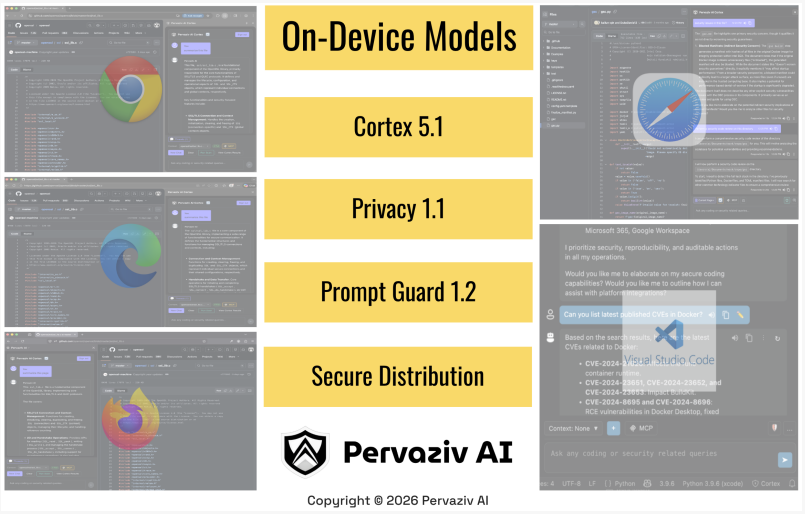
Pervaziv AI Powers On-Device Local Models in Cortex, Bringing Private, Low-Latency AI Controls to Developer Workflows

Cortex Privacy, Cortex Prompt Guard & Cortex Secure Distribution extend Pervaziv AI's model independence strategy with local AI safety for software development



Cortex On Device Models

SAN FRANCISCO, CA, UNITED STATES, July 8, 2026 /EINPresswire.com/ -- Pervaziv AI today announced a major expansion of its on-device local model strategy for Cortex, bringing private, low-latency AI controls directly into the developer experience.

The announcement follows the recent launch of Cortex 5.0 and Cortex-LLM-1.0, Pervaziv AI's first internally trained AI model for secure software development. Cortex 5.0 marked an important step toward model independence by introducing specialized AI behavior for security analysis, secure remediation, structured findings, and secure agentic engineering workflows. With on-device local models, Pervaziv AI is extending that direction into another critical layer of enterprise AI adoption: local privacy, prompt safety, and secure model distribution.

The new work centers on three capabilities:

- Cortex Privacy, released as cortex-privacy-1.1, for local sensitive-data detection and privacy-aware preflight scanning.

- Cortex Prompt Guard, released as cortex-prompt-guard-1.2, for local prompt-injection and instruction-risk classification.

- Cortex Secure Distribution, for private model delivery, versioning, integrity verification, provenance metadata, packaged product behavior, and enterprise-friendly lifecycle management.

Together, these capabilities help organizations use AI across software development with stronger privacy posture, faster local decisioning, lower operational friction, and more control over how AI safety behavior is delivered into real developer environments.

The larger message is simple: not every AI decision should require a remote model call.

AI-assisted software development is now part of everyday engineering work. Developers use AI to review code, explain issues, summarize context, generate fixes, understand logs, and move faster across complex software systems. But as AI becomes more deeply embedded in development, enterprise adoption still depends on a practical question: where does sensitive context go?

Developer context often contains far more than code. It may include credentials, tokens, private endpoints, database connection strings, cloud account identifiers, stack traces, logs, customer references, internal hostnames, and operational data. It may also include untrusted content from web pages, issue trackers, package metadata, pull request comments, documentation, generated text, and copied logs.

For enterprises, the concern is not only whether an AI model can answer a question. The concern is whether the AI system can decide what should be shared, what should be protected, what should be redacted, what should be blocked, and what should be handled locally before a larger model is invoked.

Pervaziv AI's on-device local model strategy addresses that layer of the problem.

"Cortex 5.0 moved Pervaziv AI closer to model independence by introducing specialized AI behavior for secure software development. On-device local models take that strategy one layer deeper," said Anoop Jaishankar, Founder and CEO of Pervaziv AI. "Enterprises should not have to send every sensitive prompt, code snippet, log, or browser context to a remote model just to decide whether it is safe. The future of secure AI development is layered: local models for fast privacy and safety decisions, specialized models for secure reasoning, and governed workflows that keep control where enterprises need it most."

Local AI Controls Where Developers Already Work

Pervaziv AI's on-device local model work is designed to run where Cortex users already work: inside VS Code and across major browsers, including Chrome, Safari, Edge, and Firefox.

The goal is not to ask developers to change tools. The goal is to make privacy and AI safety

controls available directly inside the surfaces where modern software work already happens.

Developers move between IDEs, code repositories, issue trackers, cloud consoles, documentation, security advisories, package registries, pull requests, browser-based AI tools, and collaboration systems. AI assistance increasingly follows that same pattern.

That creates a need for consistent local controls across the surfaces where developer context appears.

On-device local models help Cortex make high-frequency safety decisions close to the user, before sensitive content is sent anywhere else. These decisions can include whether a prompt contains sensitive data, whether untrusted content includes prompt-injection language, whether context should be redacted before a larger model receives it, or whether additional safeguards should be applied before an AI workflow continues.

This is not about replacing large reasoning models. It is about using the right model at the right layer of the workflow.

Large models remain valuable for deeper reasoning, code explanation, architecture analysis, security review, remediation planning, and agentic workflows. Smaller specialized local models are better suited for immediate preflight controls that must be fast, private, repeatable, and close to the development environment.

That layered architecture is central to Cortex.

Cortex Privacy: Sensitive-Data Detection Before Context Leaves the Client
---
Cortex Privacy, released as cortex-privacy-1.1, is focused on detecting sensitive data before it leaves the local environment.

In developer workflows, sensitive data can appear in many forms. A code snippet may include an API key. A stack trace may include a private endpoint. A log file may contain an email address, customer identifier, session token, internal hostname, or database URL. A configuration file may include secrets or environment-specific values.

Cortex Privacy is designed to identify these risks locally so Cortex clients can take the appropriate action before a broader AI workflow begins.

Those actions may include warning the user, redacting sensitive spans, blocking unsafe sharing, routing the request differently, or applying product-specific privacy behavior based on enterprise policy and workflow context.

This is a specialized ML problem. Privacy protection often requires more than a broad safe or

unsafe label. The product may need to know what span of text is sensitive, what category it belongs to, and how that span should be handled. A remote model should not have to receive sensitive content in order to decide whether the content is sensitive.

From a product perspective, the capability helps make AI assistance safer by default. Developers should not have to manually inspect every prompt, log, code snippet, configuration file, or browser context before using AI. Local privacy scanning provides a protective layer that operates inside the workflow.

From a business perspective, the value is direct: reduce the likelihood that sensitive developer, customer, operational, or enterprise data is unintentionally exposed through AI workflows.

It also supports a practical economic benefit. If sensitive content can be detected and handled locally, Cortex can avoid spending remote inference tokens on content that should never have been sent upstream.

Cortex Prompt Guard: Local Defense Against Prompt Injection

Cortex Prompt Guard, released as cortex-prompt-guard-1.2, focuses on detecting prompt-injection and instruction-manipulation attempts before they influence AI behavior.

Prompt injection can create real operational risk in AI-assisted workflows. It can attempt to make an AI assistant ignore system instructions, reveal hidden context, misuse tools, expose sensitive data, bypass policy, or take actions outside the intended workflow.

In developer environments, prompt-injection risk can appear in places that look ordinary: web pages, package READMEs, dependency descriptions, pull request comments, issue tracker entries, copied logs, or documentation pages.

Cortex Prompt Guard provides a lightweight local classifier for this risk category. Its purpose is to detect instruction-risk patterns before the content influences a larger model or agentic workflow.

The model is tuned for product behavior, not just benchmark performance. Production AI safety is not only about identifying risk. It is also about creating a reliable developer experience that knows when to warn, when to allow, and when to apply safeguards without unnecessary friction.

In security UX, overblocking matters. A control that blocks too aggressively slows teams down. A control that is too permissive fails when it matters most. Product-ready decision quality sits between those extremes.

Cortex Prompt Guard is designed for prompt-injection detection, local instruction-risk

classification, usability-aware risk reduction, and runtime compatibility across developer surfaces.

This is especially important across browser-based AI workflows, where developers move between code, documentation, cloud portals, and internal systems. A local prompt-risk classifier helps apply a consistent safety posture without requiring every check to call a remote model.

Cortex Secure Distribution: From Model Experiment to Product Capability

A model is not production-ready just because it performs well in evaluation.

For on-device local models, distribution is part of the product. The client needs to know which model version to use, how to verify it, how to apply the intended product behavior, and how to keep runtime behavior consistent across releases.

Cortex Secure Distribution addresses that layer.

Pervaziv AI's local model delivery approach is built around private, controlled distribution rather than direct runtime dependency on external model sources. Customers do not need end-user access to gated external systems during normal product use. The product can distribute approved, versioned local model capabilities through enterprise-controlled channels.

The distribution infrastructure includes stable model versions, private delivery, integrity verification metadata, provenance metadata, packaged product behavior, runtime compatibility validation, and release-level governance.

For enterprise customers, this means fewer setup requirements, fewer runtime access failures, stronger control over model availability, and more predictable AI behavior. Cortex can deliver the version that has been approved, validated, and packaged for the intended runtime environment.

For engineering teams, this creates a cleaner path from model development to product release. A local model can be evaluated, versioned, packaged, verified, distributed, loaded, tested, and rolled back if needed.

For product teams, it supports a more predictable operating model. High-frequency safety checks can run locally without incurring remote inference cost on every classification, while larger AI systems remain available for deeper tasks.

Stable Versions and Repeatable Runtime Behavior

Local AI controls need stable versions because the product experience depends on repeatability.

If privacy detection changes unexpectedly, developers may see inconsistent redaction or missed sensitive spans. If prompt-risk classification changes unexpectedly, warnings may appear or disappear without explanation. If a model package behaves differently across clients, enterprises lose confidence in the control layer.

Stable versions make it possible to test, promote, roll back, and compare model behavior over time.

Versioned local models also support governance. Teams can understand which model version made a decision, which product behavior was active, and how results can be reproduced. That traceability matters for enterprise AI adoption.

Stable versions improve ML operations by mapping evaluation results to a concrete release, validating runtime compatibility before promotion, simplifying rollback paths, and making client behavior easier to reproduce.

Private Controls, Lower Latency, and Better AI Economics

The on-device local model strategy reflects a practical economic reality: not every AI step should consume remote model tokens.

In many AI-assisted workflows, safety checks happen repeatedly. A prompt may be checked before it is sent. A file may be checked before it is attached. A web page may be checked before it is summarized. A log may be scanned before it becomes model context.

If every one of those decisions requires a remote model call, the system adds latency, cost, dependency, and privacy exposure.

Running narrow AI controls locally reduces unnecessary token consumption and reserves larger models for higher-value reasoning tasks.

Local models do not replace larger models. They help orchestrate when larger models should be used, what context they should receive, and how sensitive or risky content should be handled before remote inference begins.

For enterprises scaling AI across many developers and workflows, cost control and privacy control become part of the same architecture.

Privacy and Security as Product Infrastructure

Pervaziv AI’s broader direction is to make privacy and safety controls part of the developer experience itself.

Cortex Privacy checks for sensitive content before it leaves the client. Cortex Prompt Guard checks whether content may be trying to manipulate AI behavior. Cortex Secure Distribution ensures that local models are versioned, verified, and delivered securely.

Together, these capabilities create a stronger foundation for AI-assisted development. They do not replace larger reasoning models. They make those workflows safer, faster, more cost-efficient, and more enterprise-ready.

A large reasoning model may be better for analysis, remediation, planning, or explanation. A local classifier is often better for immediate preflight control. Combining the two creates a more practical architecture than relying on one model for every task.

This is why Pervaziv AI's on-device local model work is a natural follow-on to Cortex 5.0 and Cortex-LLM-1.0. Cortex 5.0 strengthened specialized model behavior for secure software development. On-device local models extend that philosophy into the client-side safety layer.

Enterprises need powerful AI, but they also need control. They need model flexibility, governance, developer productivity, and strong privacy and security boundaries.

Cortex is being designed around those requirements.

A New Layer in the Enterprise AI Control Layer

The on-device local model work continues Pervaziv AI's broader progression across secure coding, cybersecurity automation, privacy-aware AI, browser and IDE workflows, cloud intelligence, DevSecOps, AI security review, and secure agentic engineering.

Earlier Cortex releases expanded the platform across browsers, IDEs, cloud ecosystems, enterprise integrations, privacy scanning, threat modeling, and validation workflows. Cortex 5.0 added a specialized model foundation with Cortex-LLM-1.0. The new local model work adds another layer: private AI safety controls that operate directly inside the client.

The result is a more complete Enterprise AI Control Layer.

Instead of treating AI as a single remote assistant, Cortex is moving toward a layered system of specialized capabilities. Some capabilities run locally for privacy and speed. Some run as specialized models for secure reasoning. Others use larger models for deeper analysis.

That layered approach gives enterprises more flexibility and control, allowing the system to use the right model for the right task and keep sensitive decisions closer to the developer when local execution is the better option.

Built for the Next Phase of Secure AI Development

The first phase of AI coding adoption focused on speed and generation. Developers used AI to write code faster and accelerate routine tasks. The next phase is different.

Enterprises now need AI systems that can help teams build, secure, validate, govern, and operate software with confidence. That means AI-generated code must be reviewable, security findings must be structured, remediation must be focused, and privacy must be built into the workflow.

On-device local models are becoming an important part of that future.

They make AI controls faster because they run close to the user. They make AI controls more private because sensitive context does not need to leave the client for every decision. They make AI systems more resilient because high-frequency checks can operate with less dependency on external services. They make AI economics more practical because not every safety decision consumes remote tokens.

“Enterprises are moving from AI experimentation to AI operations,” said Jaishankar. “That shift requires more than powerful models. It requires local controls, specialized behavior, secure distribution, privacy-aware workflows, and a control layer that can manage AI across the software development lifecycle. On-device local models are a major step toward that future because they bring intelligence closer to the developer while keeping enterprise control at the center.”

Availability

Cortex Privacy cortex-privacy-1.1, Cortex Prompt Guard cortex-prompt-guard-1.2, and Cortex Secure Distribution are part of Pervaziv AI’s continuing work to expand secure AI capabilities across software engineering, privacy-aware AI workflows, DevSecOps, browser-based development, IDE-based development, and enterprise AI governance.

The capabilities strengthen the company’s long-term roadmap for model independence, on-device local inference, specialized AI security behavior, prompt-injection defense, privacy protection, structured safety decisions, secure model distribution, and enterprise-grade secure agentic engineering.

Pervaziv AI
Pervaziv AI
email us here
Visit us on social media:
LinkedIn
Instagram

YouTube
X
Other