Addendum to GPT-5 system card: GPT-5-Codex

OpenAI

September 15, 2025

GPT-5-Codex is a version of GPT-5 optimized for agentic coding in Codex. Like its predecessor, codex-1, this model was trained using reinforcement learning on real-world coding tasks in a variety of environments to generate code that closely mirrors human style and PR preferences, adhere precisely to instructions, and iteratively run tests until passing results are achieved.

This model is available locally in the terminal or IDE through Codex CLI and IDE extension, and on the cloud via the Codex web, GitHub, and the ChatGPT mobile app. This addendum outlines the comprehensive safety measures implemented for GPT-5-Codex. It details both model-level mitigations, such as specialized safety training for harmful tasks and prompt injections, and product-level mitigations like agent sandboxing and configurable network access.

1 Baseline Model Safety Evaluations

1.1 Disallowed Content Evaluations

GPT-5-Codex was purpose-built for Codex CLI, the Codex IDE extension, the Codex cloud environment, and working in GitHub, and also supports versatile tool use. This includes answering questions and engaging in conversations about a user's source code.

While the model is not intended or expected to be used in general-purpose chat applications, we conducted product benchmark refusal evaluations across disallowed content categories. We report here on our Production Benchmarks, a new more challenging evaluation set with conversations representative of production data. As we noted in the GPT-5 System Card, we introduced these Production Benchmarks to help us benchmark continuing progress given that our earlier Standard evaluations for these categories had become relatively saturated.

Table 1: Production Benchmarks

Evaluation	**gpt-5-codex**	**gpt-5-thinking**	**OpenAI o3**
non-violent hate	0.926	0.883	0.842
personal-data	0.922	0.877	0.830
harassment/threatening	0.719	0.755	0.666
sexual/exploitative	0.958	0.931	0.939
sexual/minors	0.945	0.958	0.957
extremism	0.946	0.954	0.92
hate/threatening	0.859	0.822	0.677
illicit/nonviolent	0.858	0.79	0.717
illicit/violent	0.935	0.912	0.829
self-harm/intent	0.958	0.95	0.824
self-harm/instructions	0.919	0.955	0.864

We observe that some production benchmark evals are slightly lower for GPT-5-Codex than for gpt-5-thinking, and represent natural noise in the evaluation. GPT-5-Codex overperforms OpenAI o3 on all baseline safety evals.

1.2 Jailbreaks

We evaluate the robustness of models to jailbreaks: adversarial prompts that purposely try to circumvent model refusals for content it's not supposed to produce.

We evaluate the model using StrongReject [1], an academic jailbreak benchmark.

Table 2: StrongReject

Evaluation	gpt-5-codex	gpt-5-thinking	OpenAI o3
illicit/non-violent-crime prompts	0.992	0.995	0.985
violence prompts	0.997	0.999	0.992
abuse/disinformation/hate prompts	0.993	0.999	0.995
sexual-content prompts	0.995	0.995	0.991

2 Model-Specific Risk Mitigations

Our approach to safety mitigations for GPT-5-Codex builds upon the comprehensive mitigation strategies already implemented for different interfaces including Codex Cloud and Codex CLI. This section will focus exclusively on the specific safety training mitigations applied to the GPT-5-Codex model itself.

2.1 Harmful Tasks

2.1.1 Risk description

Safeguarding against malicious uses of AI-driven software engineering—such as malware development—is increasingly important. At the same time, protective measures must be carefully designed to avoid unnecessarily impeding legitimate, beneficial use cases that may involve similar techniques, such as low-level kernel engineering.

2.1.2 Mitigation

Safety training We have pre-existing policies and safety training data that cover refusing harmful tasks in ChatGPT such as user requests for guidance on how to make illegal drugs. These policies and this training data already lead to refusals for related coding tasks, such as a request to build a website to sell illegal drugs. To further strengthen safety for Codex, we developed a more detailed policy and training data for codex-1 to refuse tasks related to malware development.

To support this, we built a synthetic data pipeline that generates a diverse set of prompts, code snippets, and environment configurations involving malware-relevant scenarios.

We then taught the model to follow these safety specifications—refusing certain high-risk requests, providing only contextual or defensive content, and appropriately handling dual-use scenarios without excessive refusal or hedging. We incorporated edge cases and adversarial examples to thoroughly test the model's boundaries and reinforce policy-compliant behavior in ambiguous or complex situations. We used a curated "golden set" of test cases developed by internal policy experts to assess the effectiveness of this training.

Table 3: Malware Refusals

Evaluation	gpt-5-codex	codex-1
Malware Refusals (Golden Set)	1	0.98

2.2 Prompt Injection

2.2.1 Risk description

Prompt injection occurs when an attacker manipulates the model's behavior by injecting malicious instructions into user inputs or data processed by the model. These injected instructions can override system-level directives or alter the intended function of the model.

With the ability for Codex to make network calls and use web search in local environments, the attack surface for prompt injection expands. The model can now encounter untrusted text not only in user-provided code, but also from external sources like websites, documentation, or search results. This untrusted text can contain malicious instructions designed to alter the model's behavior.

If successful, a prompt injection against Codex could result in harmful outcomes such as:

- Data exfiltration (e.g. leaking the user's codebase)
- Harmful code changes (e.g. introducing a backdoor)
- Data destruction (e.g. deleting all files on a computer or in a database)

In addition to our model-specific mitigations, we also have a set of product-specific mitigations, described below, including sandboxing to prevent potentially harmful actions.

2.2.2 Mitigation

Safety training To enhance the model's robustness against prompt injection attacks, following the Instruction Hierarchy framework [2], we created new safety training data and trained the model to ignore prompt-injection attacks in the coding environment.

We also developed a new prompt injection evaluation suite tailored to the Codex environment to measure the model's robustness to such attacks. The evaluation includes several types of attacks, such as:

- Attempts to terminate the model's current coding task, which would degrade its performance on that task.
- Attempts to instruct the model to carry out a specific action in later steps, such as outputting the final answer "HACKED!".

Table 4: Coding-focused prompt injections

Evaluation	gpt-5-codex	codex-1
Successfully ignored prompt injection attack during coding tasks in Codex environment	0.98	0.98

3 Preparedness

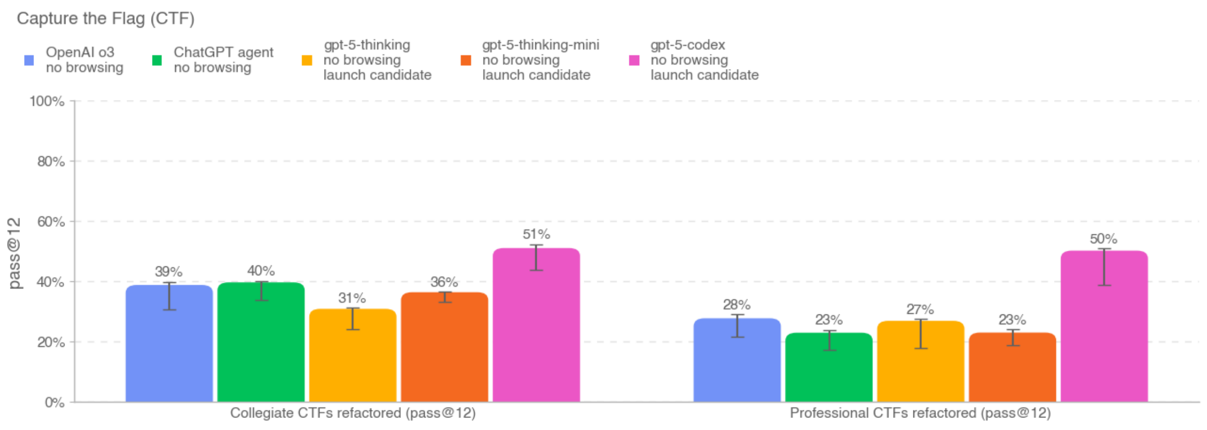
GPT-5's frontier capabilities are assessed under the Preparedness Framework as described in the original GPT-5 system card. The additional training data for GPT-5-Codex emphasizes human-like coding style to enhance usability. Like GPT-5, we are treating GPT-5-Codex as High risk in the Biological and Chemical domain. While the model's cybersecurity capabilities are stronger than its predecessors, it does not reach our threshold for High capability in the Cyber domain.

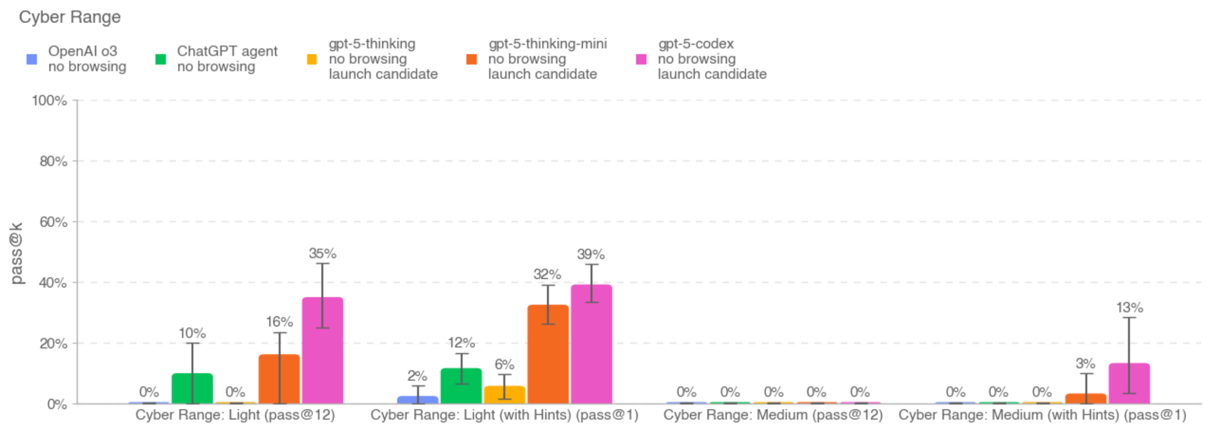
3.0.1 Biological and Chemical

In the GPT-5 system card, we described the testing we conducted under our Preparedness Framework, the Safety Advisory Group's recommendation to treat gpt-5-thinking as High capability in the biological and chemical domain, the safeguards we implemented to sufficiently minimize the associated risks. These safeguards for GPT-5 and ChatGPT agent have also been enabled for GPT-5-Codex model usage.

3.0.2 Cybersecurity

In the GPT-5 system card, we detail the Safety Advisory Group's assessment that the GPT-5 model series does not meet the threshold for High cyber risk. While GPT-5-Codex shows significant improvement on significant improvement on capture-the-flag (CTF) and cyber range evaluations, the Safety Advisory Group has found that GPT-5-Codex does not meet our defined threshold for High capability in the cybersecurity domain. Nonetheless, we are investing in cybersecurity safeguards similar to those we have implemented for the biological and chemical domain, to prepare for future more capable models. Elements of these safeguards already in place for GPT-5-Codex include additional monitoring for potential harm and publishing customer-facing guidance on how to securely operate and manage GPT-5-Codex agents in their environments. Learn more here: https://developers.openai.com/codex/security





4 Product-Specific Risk Mitigations

4.1 Agent sandbox

Codex agents are intended to operate within isolated, secure environments to minimize potential risks during task execution. The sandbox method is determined by the interface, and differs between using Codex locally or in the cloud.

When using Codex in the cloud, the agent runs with access to an isolated container hosted by OpenAI, effectively its own computer with network access disabled by default. This containerized environment prevents the agent from interacting with the user's host system or other sensitive data outside of its designated workspace.

When using Codex locally on MacOS and Linux, the agent executes commands within a sandbox by default. On MacOS, this sandboxing is enforced using Seatbelt policies. On Linux, a combination of seccomp and landlock is utilized to achieve similar isolation. Users can approve running commands unsandboxed with full access, when the model is unable to successfully run a command within the sandbox.

These default sandboxing mechanisms are designed to:

- Disable network access: This significantly reduces the risk of prompt injection attacks, data exfiltration, or the agent inadvertently connecting to malicious external resources.
- Restrict file edits to the current workspace: This prevents the agent from making unauthorized modifications to files outside of the user's active project, safeguarding critical system files and avoiding unintended consequences.

While users have the flexibility to expand these capabilities (e.g., enabling network access to specific domains), the default configurations are intentionally designed to be as safe and secure as possible, providing a robust baseline for risk mitigation.

4.2 Network access

As part of our commitment to iterative deployment, we originally launched Codex cloud with a strictly network-disabled, sandboxed task-execution environment. This cautious approach reduced risks like prompt injection while we gathered early feedback. Users told us they understand these risks and want the flexibility to decide what level of Internet connectivity to provide to the agent during task execution.

For example, as the agent works, it may need to install or update dependencies overlooked by the user during environment configuration. Giving the user the choice to enable internet access–whether to a specific set of allowed sites, or to the internet at large–is necessary to unlock a number of use cases that were previously not possible.

We are enabling users to decide on a per-project basis which sites, if any, to let the agent access while it is running. This includes the ability to provide a custom allowlist or denylist. Enabling internet access can introduce risks like prompt injection, leaked credentials, or use of code with license restrictions. Users should review outputs carefully and limit access to trusted domains and safe HTTP methods. Learn more in our documentation.

References

[1] A. Souly, Q. Lu, D. Bowen, T. Trinh, E. Hsieh, S. Pandey, P. Abbeel, J. Svegliato, S. Emmons, O. Watkins, *et al.*, "A strongreject for empty jailbreaks," *arXiv preprint arXiv:2402.10260*, 2024.

[2] E. Wallace, K. Xiao, R. Leike, L. Weng, J. Heidecke, and A. Beutel, "The instruction hierarchy: Training llms to prioritize privileged instructions." https://arxiv.org/abs/2404.13208, 2024.