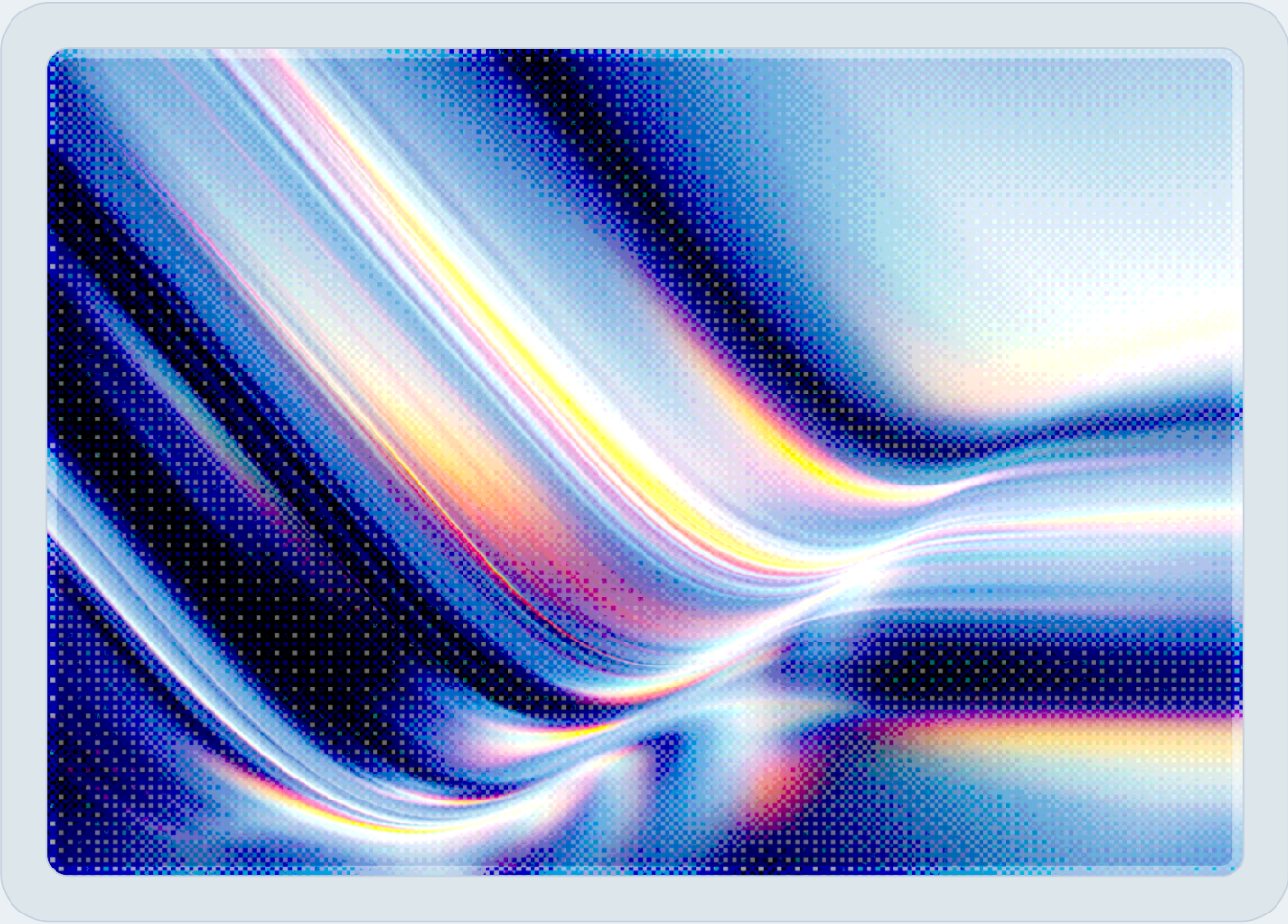
CSAI Foundation | Cloud Security Alliance

VS Code Zero-Day: One-Click GitHub Token Theft

Webview Sandbox Escape Exposes Full Repository Access in AI Developer Toolchains

2026-06-04

AI-assisted Rapid Research



CSAI

cloud security alliance®

Key Takeaways

- Security researcher Ammar Askar publicly disclosed a zero-day vulnerability in github.dev, GitHub's browser-hosted Visual Studio Code editor, on June 2, 2026, demonstrating that a single malicious link click is sufficient to steal a victim's full-access GitHub OAuth token [1][2].
- The attack chains five VS Code behaviors—webview keydown event bubbling, Jupyter notebook HTML execution, notification primary-action triggering, local workspace extension trust bypass, and custom keybinding command execution—requiring no user interaction beyond opening the crafted link [1].
- The stolen OAuth token grants read and write access to every repository the victim can access, not merely the repository that was opened, because github.dev receives a broadly scoped session token at load time rather than a per-repository credential [1][3].
- Microsoft applied two stopgap patches on June 3, 2026, one day after public disclosure: blocking `skipPublisherTrust` bypass via commands and preventing `keydown` events from bubbling out of notebook webviews; the patches address the specific proof-of-concept exploit, though the scope of remaining attack surface within VS Code's webview messaging architecture has not been publicly confirmed by Microsoft or independent researchers as of this writing [2][4].
- Askar released the full proof-of-concept with one hour of advance notice to GitHub security rather than through Microsoft's Security Response Center, citing a prior incident in which MSRC silently patched a different VS Code vulnerability he reported without attribution or acknowledgment of security impact [1][5].
- The vulnerability is directly relevant to teams using AI coding assistants built on the VS Code platform—including GitHub Copilot and agentic coding tools—because these environments rely on the same GitHub credential fabric and could inherit compromised access if a developer's token were stolen via this mechanism [6][7].

Background

GitHub.dev is Microsoft's browser-hosted implementation of Visual Studio Code, accessible at `github.dev`, that lets developers browse and edit repositories without installing any local software. When a user navigates from `github.com` to `github.dev` on any repository, GitHub's platform automatically posts a full OAuth session token to the editor. This design provides seamless authentication but carries a significant security implication: the token is scoped to all repositories the user can access, not just the one opened. A developer with access to hundreds of organizational private repositories therefore transmits a single credential that unlocks the entire corpus the moment the editor loads [1][3].

Ammar Askar, a security researcher who had previously engaged Microsoft's Security Response Center over a related VS Code vulnerability, discovered that VS Code's webview sandbox could be escaped through a chain of individually documented behaviors that, when composed, allowed untrusted JavaScript running in a sandboxed context to install a malicious editor extension and exfiltrate the pre-loaded OAuth token—all without requiring any user action beyond clicking a link. The exploit completes in approximately thirty seconds or less based on the proof-of-concept demonstration, though secondary reporting has placed the figure at under one minute; the precise timing requires verification against the original source [1].

Askar published the full proof-of-concept on June 2, 2026, on the VS Code issue tracker and his personal blog simultaneously, notifying a GitHub security contact one hour before posting rather than following the ninety-day coordinated disclosure period championed by Google Project Zero and widely adopted across major bug bounty and vulnerability research programs. His stated rationale centered on a previous disclosure to MSRC in which the agency "silently fixed the bug I pointed out without any credit" and classified it as having "no security impact." Askar has since stated he will "go with the public disclosure route for any VSCode-related bugs" [1][5]. Microsoft issued patches the following day, June 3, and stated that "the issue has been mitigated for our services and no customer action is required" [2][4].

The vulnerability does not affect the VS Code desktop application when used with local repositories, though the desktop version with a remote workspace presents a related but distinct attack surface. The github.dev web editor is the primary affected environment [2][3].

Security Analysis

The Five-Stage Exploit Chain

The attack is a composition of five VS Code behaviors, none of which individually constitutes a critical flaw but which together create a complete credential-theft pipeline.

Stage one exploits the fact that VS Code's webview component posts `keydown` events to the parent window via `"did-keydown"` messages. This feature exists to enable keyboard navigation within embedded webviews, but it means that untrusted JavaScript executing inside a sandboxed iframe can synthesize keyboard events that are treated by the main editor as genuine user input. The synthesized events are indistinguishable, at the application layer, from physical keystrokes.

Stage two uses VS Code's Jupyter notebook renderer to execute arbitrary JavaScript. Markdown cells within `.ipynb` files can contain HTML, and that HTML can include an `<img>` tag with an `onerror` handler: `<img src="data:foobar" onerror="attackerJavaScript();">`. When VS Code renders the notebook, it executes the onerror handler within the webview's JavaScript context, providing the attacker a reliable code execution point inside the sandboxed environment.

Stage three leverages VS Code's notification system. When a repository contains a `.vscode/extensions.json` file recommending an extension, VS Code displays a notification asking the user to install it. The attacker's JavaScript payload waits ten seconds for this notification to appear, then dispatches a synthetic `Ctrl+Shift+A` keyboard event—the default binding for "Notifications: Accept Notification Primary Action." This accepts the extension installation prompt without any visible user confirmation in the UI.

Stage four exploits VS Code's handling of local workspace extensions. Extensions placed in the `.vscode/extensions/` directory of a trusted workspace bypass the publisher trust dialog that would normally alert users to an unfamiliar publisher identity. The workspace's trusted status acts as the trust gate, and because the attacker controls the repository layout, they control both the Jupyter notebook payload and the local extension it installs.

Stage five uses the newly installed extension's contributed keybindings to trigger the final action. The extension adds a custom keybinding—bound to `Ctrl+F1`—that invokes the VS Code command `workbench.extensions.installExtension` with the `skipPublisherTrust: true` context. The Jupyter notebook payload fires a second synthetic keypress to trigger this binding five hundred milliseconds after the local extension activates. The extension then reads the pre-loaded

GitHub OAuth token from the editor's credential store and queries `https://api.github.com/user/repos` to enumerate all private repositories accessible to the victim before exfiltrating the token [1].

The complete attack requires that an attacker control a repository containing a malicious `.ipynb` notebook and a `.vscode/extensions/` directory with a prepared extension. Distributing a github.dev link to that repository—for example, via a phishing message, a social engineering campaign, or an open-source contribution invitation—is the only delivery mechanism needed. No vulnerability in the victim's operating system, browser, or network is required [1][2].

Token Scope and Blast Radius

The severity of this vulnerability is amplified by a design characteristic of github.dev's authentication model. When GitHub posts the OAuth token to the editor session, it does not scope the token to the specific repository the user opened. The token grants read and write access to every repository the authenticated user can reach, including private repositories across every organization they belong to [1][3]. An attacker who captures the token from a developer with broad organizational access therefore gains the ability to read proprietary source code, push commits to production branches, modify CI/CD configuration files, and access any secrets that were accidentally committed to repository history—and potentially without triggering authentication alerts in organizations that rely on identity-based rather than behavioral detection, since the operations proceed under the legitimate user's identity. Organizations with behavioral monitoring enabled may still detect bulk enumeration or unusual access patterns even when a legitimate token is in use.

This design trade-off optimizes for developer convenience at the cost of a large attack surface. The least-privilege alternative—issuing a per-repository, per-session token—would require additional round-trips on each repository transition but would contain the blast radius of any single token compromise to a single repository. GitHub Codespaces, by contrast, issues per-session, container-isolated credentials that are significantly more narrowly scoped than the github.dev OAuth token [3].

AI Toolchain Exposure

The vulnerability's implications extend directly into the AI-assisted development ecosystem. GitHub Copilot, Microsoft's AI pair-programming service, operates within the VS Code environment and authenticates through the same GitHub identity infrastructure. An attacker who steals a developer's GitHub OAuth token via this mechanism does not gain direct control of Copilot's model, but they do gain the ability to manipulate the repositories that Copilot indexes as context for the affected developer.

Malicious code pushed to those repositories could influence AI-generated suggestions the next time the victim or their colleagues use Copilot against that codebase—an indirect prompt-poisoning vector worth tracking as AI coding assistants become more autonomous.

More directly, organizations running agentic AI coding tools that clone repositories, execute code in development environments, and authenticate using GitHub credentials would expose those agent sessions to full compromise if the developer credentials used to authorize them are stolen. The RoguePilot vulnerability disclosed in February 2026, in which a malicious GitHub issue could trigger Copilot to exfiltrate the `GITHUB_TOKEN` from a Codespace [6], demonstrated that AI coding agents operating in GitHub-authenticated environments are already a target class. The June 2026 VS Code zero-day represents a parallel attack path: instead of attacking the AI agent directly, an adversary compromises the human developer's credential and inherits access to everything the agent would have accessed on the developer's behalf.

The broader pattern is consistent with the Nx Console supply chain attack disclosed in May 2026, in which a compromised VS Code Marketplace extension installed by a GitHub employee led to the exfiltration of approximately 3,800 internal GitHub repositories [7]. In that incident, the delivery mechanism was a trusted extension rather than a crafted github.dev link, but the outcome was structurally similar: in both cases a compromised developer credential yielded broad repository access, though the delivery mechanisms and affected populations differed significantly. Together, these incidents suggest that the VS Code extension and webview ecosystem is increasingly targeted by adversaries seeking developer repository credentials at scale.

Disclosure Process and Structural Risk

Askar's decision to forgo full coordinated disclosure reflects a structural tension in enterprise security research. MSRC's handling of prior VS Code vulnerability reports—in which a reported flaw was patched without researcher credit or classification as a security issue—creates an incentive for researchers to seek public accountability through full disclosure rather than trusting a vendor's internal process. The one-hour notice Askar provided is not consistent with industry standard responsible disclosure norms, which generally call for a minimum ninety-day window before public release. However, the rapid deployment of patches within twenty-four hours of public disclosure suggests that Microsoft had the technical capability to respond quickly when motivated by public visibility [2][4].

The episode also illustrates that full-disclosure pressure appears to remain a significant forcing function for timely vendor response to developer toolchain vulnerabilities. Organizations that rely on github.dev in their development workflows have limited visibility into the vulnerability disclosure negotiations between security researchers and platform vendors, and must therefore maintain controls at the toolchain and credential layer regardless of vendor patch timelines.

Recommendations

Immediate Actions

Organizations should take the following steps in the near term to reduce exposure from this and structurally similar developer toolchain credential threats.

First, confirm that all github.dev sessions have been updated to the patched version. Microsoft stated on June 3 that mitigations were applied server-side, meaning that users who access github.dev after that date receive the patched implementation without any local action required. However, teams should audit whether any active sessions predating June 3, 2026, remain cached and could carry unpatched state.

Second, audit GitHub OAuth token scopes in use across the organization. For services and automation that require GitHub API access, replace broad personal access tokens with fine-grained tokens scoped to specific repositories and operations. GitHub's fine-grained personal access token feature, available since 2022, allows tokens to be restricted to a subset of repositories and a subset of permissions, dramatically reducing blast radius if a token is compromised [13].

Third, enable GitHub's push protection and secret scanning features on all organizational repositories. If an attacker uses a stolen token to push code or exfiltrate secrets, these controls create detection opportunities that would otherwise be absent.

Short-Term Mitigations

Organizations should review their policies for accessing repositories via github.dev links from untrusted sources. Teams that do not have a use case for the web-based editor can disable access to `github.dev` at the network level or through browser policy, reducing their attack surface while the broader webview sandbox architecture in VS Code is hardened. Users who do access github.dev regularly should clear browser site data for the domain after each session, which removes the locally cached token and forces re-authentication; they should also review active GitHub sessions at github.com/settings/sessions and revoke any sessions no longer needed, since server-side session invalidation requires action in the GitHub account settings rather than the browser alone.

Developer security training should be updated to include awareness of crafted github.dev links as a phishing vector. The social engineering required for this attack is low: an invitation to review a pull request, an open-source collaboration request, or a link embedded in a security advisory can all plausibly

direct a developer to a malicious repository. The technical sophistication of the exploit makes user training insufficient as a sole control, but raising awareness of the attack pattern reduces the likelihood that social engineering succeeds in delivering the link.

For teams using AI coding assistants that authenticate through GitHub credentials, apply the same token-scoping discipline to the service accounts used by those tools. An AI agent that only needs read access to a specific set of repositories should authenticate with a token that can only grant that access, so that compromise of the agent's credential does not expose organizational repositories broadly.

Strategic Considerations

The structural vulnerability underlying this attack—that github.dev's session token grants access to all repositories regardless of which one was opened—is a platform design decision that Microsoft should revisit. CSA encourages organizations to engage with GitHub on this issue directly and to track whether the June 3 patches address only the specific proof-of-concept or whether they include changes to token scoping behavior.

More broadly, the VS Code webview architecture presents an ongoing challenge for secure developer tooling. The postMessage API that enables rich extension capabilities also creates a permeable boundary between sandboxed content and the main editor environment. As VS Code becomes the runtime for AI coding agents, agentic automation, and remote development environments—not merely a text editor—the security implications of this architecture warrant dedicated threat modeling. Organizations evaluating developer toolchain security posture should include VS Code's webview sandbox boundaries in their attack surface analysis alongside traditional endpoint and network controls.

The pattern of AI developer tools becoming credential theft vectors—seen across the RoguePilot Codespaces vulnerability in February 2026, the OpenAI Codex command injection vulnerability in March 2026, and now the github.dev zero-day in June 2026 [6][8]—indicates that adversaries have identified the AI coding toolchain as a high-yield target. The credential, not the AI model, is the primary objective. Organizations should assume that any tool operating in an authenticated developer environment is a potential theft vector and apply accordingly strict identity hygiene, credential scoping, and behavioral monitoring.

CSA Resource Alignment

This incident maps directly to several active areas of CSA guidance and framework development.

MAESTRO (Multi-Agent Environments and Security Threat Representation and Operationalization): The MAESTRO threat modeling framework for agentic AI systems identifies supply chain compromise of AI tooling as a Tier 5 risk at the infrastructure and integration layer, noting that AI agents and pipelines relying on third-party libraries and platform components inherit the security posture of those dependencies [12]. The github.dev zero-day represents a Tier 5 compromise path: an attacker who gains a developer's GitHub OAuth token via the web editor can tamper with the repositories that agentic AI systems use as their primary context source, propagating the compromise upward through the trust hierarchy without ever directly attacking the AI model or its infrastructure. MAESTRO's recommended controls—runtime environment isolation, credential scope minimization, and behavioral monitoring of agent actions—apply directly to mitigating this class of threat.

AI Controls Matrix (AICM): CSA's AICM [11], which builds on and extends the Cloud Controls Matrix to provide 243 control objectives across 18 AI-specific domains, addresses the developer toolchain and supply chain dimensions implicated in this incident. The Supply Chain Management domain provides applicable controls for managing the trustworthiness of development environment components, extension registries, and third-party tools used by AI development teams. The Identity and Access Management domain governs credential scoping, session lifecycle management, and token lifecycle controls—all directly relevant to the github.dev token architecture. Organizations that have implemented AICM controls should verify that their supply chain and identity management controls explicitly cover the developer workstation and web-based IDE attack surface.

Zero Trust Guidance: CSA's Zero Trust Architecture guidance applies the principle of least privilege to the credential design decisions exposed by this incident. A Zero Trust posture treats every authentication token as a potential attack vector, applies the narrowest possible scope to each token, and validates tokens at each resource boundary rather than trusting broad session credentials implicitly. GitHub's broad-scope github.dev token design is inconsistent with Zero Trust principles; organizations should compensate by ensuring that their fine-grained token policies and session management controls embody Zero Trust discipline even where the platform default does not.

Software Supply Chain Guidance: This incident reinforces findings from CSA's prior research on developer supply chain attacks, including the TeamPCP cascading supply chain campaigns documented in March and April 2026 [9][10]. The attack vector—a malicious repository artifact that executes on load in a trusted development environment—is structurally similar to the poisoned VS Code extension attacks used in the Nx Console incident and the TeamPCP campaign series. Organizations should extend their software supply chain controls to include the web-based IDE layer and treat crafted github.dev repository links with the same scrutiny as untrusted packages.

References

[1] Ammar Askar. "1-Click GitHub Token Stealing via a VSCode Bug." Ammar's Blog, June 2, 2026.

[2] BleepingComputer. "VS Code zero-day lets hackers steal GitHub tokens in one click." BleepingComputer, June 3, 2026.

[3] The Hacker News. "One-Click GitHub Dev Attack Lets Attackers Steal Full GitHub OAuth Tokens." The Hacker News, June 2026.

[4] SecurityWeek. "VS Code Vulnerability Allows One-Click GitHub Token Theft." SecurityWeek, June 2026.

[5] The Register. "Another bug hunter leaks Microsoft exploits in defiance of company's handling of vulnerability disclosures." The Register, June 3, 2026.

[6] The Hacker News. "RoguePilot Flaw in GitHub Codespaces Enabled Copilot to Leak GITHUB_TOKEN." The Hacker News, February 2026.

[7] VentureBeat. "GitHub confirms 3,800 internal repos stolen through poisoned VS Code extension as supply chain worm hits Microsoft's Python SDK." VentureBeat, May 2026.

[8] SiliconANGLE. "OpenAI Codex vulnerability enabled GitHub token theft via command injection, report finds." SiliconANGLE, March 30, 2026.

[9] Cloud Security Alliance. "TeamPCP: Cascading Supply Chain Attack on AI/ML Tooling." CSA AI Safety Initiative, March 2026.

[10] Cloud Security Alliance. "TeamPCP Supply Chain Cascade: When Security Tools Become Attack Infrastructure." CSA AI Safety Initiative, April 2026.

[11] Cloud Security Alliance. "AI Controls Matrix." CSA, 2025.

[12] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." CSA Blog, February 2025.

[13] GitHub. "Introducing fine-grained personal access tokens." GitHub Changelog, October 18, 2022.

