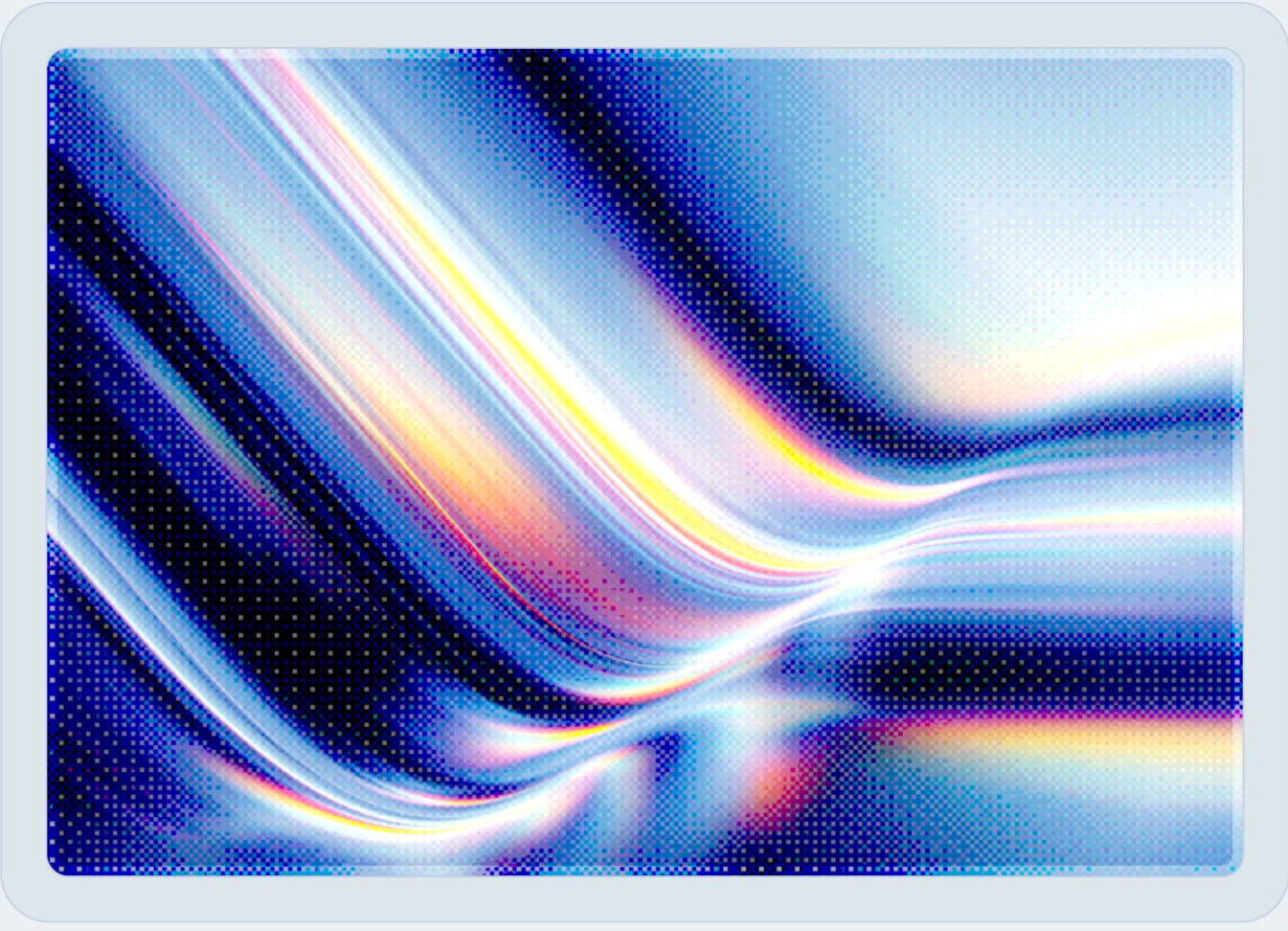
CSAI Foundation | Cloud Security Alliance

OAuth Consent Phishing: EvilTokens Bypasses MFA, Steals AI API Tokens

How Consent-Layer Attacks Sidestep Modern Defenses in Agentic AI Environments

2026-05-21

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- OAuth consent phishing and device code phishing attacks bypass multi-factor authentication entirely—victims complete a genuine MFA challenge on the attacker's behalf, and the resulting refresh tokens persist even after the victim's password is reset.
- The EvilTokens Phishing-as-a-Service platform, which surfaced in mid-February 2026, compromised more than 340 Microsoft 365 organizations across five countries within weeks of launch, spanning healthcare, financial services, manufacturing, and government [1][5].
- ConsentFix, first documented by Push Security in December 2025 and evolved to version 3 by May 2026, automates OAuth abuse against Microsoft Azure and Entra ID; when a victim already holds an active browser session, the full exploit chain executes silently with no password prompt and no MFA challenge [2].
- AI API OAuth tokens—including those used by developer tooling such as Claude Code—are now explicitly targeted; researchers demonstrated in April 2026 that malicious npm packages can silently redirect and harvest these tokens, maintaining persistence through credential rotations [3].
- The April 2026 Vercel breach illustrates the supply chain dimension: a single abandoned OAuth grant from a shadow AI application became the pivot point into Vercel's internal infrastructure, developer secrets, and downstream customer environments [4].

Background

OAuth 2.0 was designed to let users delegate specific access rights to third-party applications without handing over their credentials. The device authorization grant (RFC 8628), originally intended for input-constrained devices such as smart televisions and IoT sensors, extends this model by asking the user to visit a URL on a second device and enter a short alphanumeric code. What was an ergonomic accommodation for constrained hardware has become, in practice, a durable authentication bypass route: the legitimate OAuth flow is structurally identical to the attacker-controlled version, and the endpoint the victim visits—often microsoft.com/devicelogin—is genuine.

The consent phishing variant operates through a slightly different path, targeting the OAuth authorization code grant rather than the device code flow. An attacker registers a plausible-seeming application, crafts a consent URL requesting broad delegated permissions, and sends it to a target. The victim authenticates normally, completes any MFA step, and clicks "Accept" on the Microsoft or Google consent prompt. The attacker's backend receives a valid authorization code, exchanges it for a long-lived refresh token, and acquires persistent access to the victim's mailbox, cloud storage, calendar, and any other resource within the granted scope—all without ever possessing the user's password.

Multifactor authentication was not designed to defend against this class of attack because the victim is not being deceived about their authentication event. They are executing a real login. The deception occurs entirely at the consent layer: the victim does not recognize that "Accept" on the OAuth consent screen is the security-critical action, not the password entry or the MFA confirmation that preceded it. This cognitive gap is the foundation the entire attack class exploits.

The CSA AI Safety Initiative documented the initial EvilTokens campaign and its device code mechanics in a March 2026 research note [5]. The present note builds on that analysis to address three developments that have emerged since: the expansion of the technique via ConsentFix v3's automation, the discovery of active targeting of AI developer tooling and API credentials, and the supply chain propagation pathway illustrated by the Vercel breach.

Security Analysis

The EvilTokens Campaign and the Commoditization of Consent Phishing

From the time it surfaced in mid-February 2026, the EvilTokens kit was designed to commoditize credential theft at enterprise scale. The platform's modular pricing structure—documented in security research on the kit—placed the capability within reach of moderately resourced criminal actors, contributing to its rapid adoption [1]. Three product tiers covered the full attack chain: phishing link delivery, credential harvesting via device code, and automated post-compromise email operations through a built-in "MailVault" Outlook-clone interface.

The technical sophistication of EvilTokens lies in how it defeats the one natural defense that device code flows offer: token expiration. Standard OAuth device codes expire within a short window, and earlier device code phishing campaigns required pre-staging those codes before sending lures. EvilTokens circumvents this by keeping device codes dormant in the backend until the moment of victim interaction. When a target clicks the phishing link, a Node.js backend process requests a fresh code from the identity

provider, starting the expiration clock only when a human is present and likely to complete the flow [1]. This dynamic code generation maintains a near-perfect conversion window regardless of how slowly a victim works through the lure.

AI-generated lure content compounds the effectiveness. The platform uses generative models to produce messages tailored to each victim's role, organization, and professional context. A construction project manager receives a lure framed around subcontractor document sharing; a financial services analyst receives one referencing a counterparty compliance request. The platform's infrastructure—distributing polling operations across thousands of short-lived Railway.com nodes—further frustrates IP-based detection and blocklisting [1].

ConsentFix v3: Automation Meets Silent Exploitation

While EvilTokens operates through the device code flow, ConsentFix v3 targets the standard OAuth authorization code grant and introduces session reuse as a mechanism that eliminates the password prompt and MFA challenge even for victims with active corporate sessions—a capability that earlier device code phishing tools lacked. The technique was first documented by Push Security in December 2025 as a method for tricking victims into pasting a localhost URL containing an OAuth authorization code into an attacker-controlled page [2]. Version 2 replaced the copy-paste step with drag-and-drop to lower friction. ConsentFix v3 eliminates victim action almost entirely.

The key insight in v3 is session reuse. Microsoft Entra ID, like most enterprise identity providers, honors active browser sessions when evaluating an authorization request. If the victim already has an authenticated Microsoft session—which is nearly universal on a managed corporate device during business hours—the OAuth authorization code flow can complete without a fresh password prompt and without a new MFA challenge. The attacker-controlled page simply initiates the authorization request, the browser's existing session satisfies it, and the authorization code lands in the attacker's backend with no visible security event [2].

ConsentFix v3 now includes an automated token exchange pipeline, removing the manual steps that previously limited scale. The technique has been adopted by both state-sponsored and financially motivated threat actors targeting government and enterprise Microsoft tenants [2]. Consistent with the broader proliferation pattern documented by Push Security [2], the approach has propagated across criminal forums to reach a wide range of threat actor categories. Proofpoint tracked multiple concurrent device code and consent phishing clusters against Microsoft 365 environments throughout the first quarter of 2026, including a financially motivated actor designated TA4903 that expanded into device code phishing techniques by March 2026 [6].

AI Developer Tooling as a High-Value Target

The expansion of this attack class into AI developer tooling represents a qualitatively different risk exposure. Refresh tokens stolen from a knowledge worker's M365 account yield mailbox access and cloud file storage. Refresh tokens stolen from an AI developer's environment yield something potentially more consequential: persistent access to AI APIs, code repositories, infrastructure provisioning credentials, and the ability to inject malicious inputs into AI agent workflows.

Mitiga Labs researchers disclosed in April 2026 that Claude Code—Anthropic's AI-assisted development environment—stores its OAuth tokens and MCP (Model Context Protocol) configuration in a local file, `~/.claude.json` [3]. An attacker who can deliver a malicious npm package to a target developer's machine can register a lifecycle hook that silently redirects MCP traffic through attacker-controlled infrastructure. Whenever Claude Code initiates or refreshes an MCP session, the OAuth token transits to the attacker. If the developer rotates the token, the hook rewrites the configuration on next load, maintaining persistence. If the developer edits the MCP server URL, the hook restores it. Mitiga reported these findings to Anthropic on April 10, 2026; Anthropic characterized the issue as out of scope on April 12 [3].

The structural exposure extends beyond any single product. AI agent frameworks—as the Mitiga research illustrates—commonly store credentials in configuration files or environment variables with no hardware binding to the originating device [3][7]. A refresh token stolen through any of the consent phishing vectors described above can be replayed from attacker infrastructure with no indication that the originating device or user context has changed. The Beyond Identity research group has characterized this as the absence of hardware-bound identity in AI agent credentials—there is no cryptographic proof that a given API call originates from the legitimate agent and not from token-replay [7].

OAuth Sprawl and the Supply Chain Dimension

The April 2026 Vercel breach illustrates how OAuth's persistence properties can transform a single employee's abandoned application trial into an organizational exposure that identity governance practices did not detect or prevent. The breach originated with an employee who had trialed Context.ai, a consumer-grade AI office suite, months before the incident. The employee's Google Workspace account had granted the application an OAuth token; the employee abandoned the trial; and the OAuth grant persisted invisibly in the tenant, outlasting any memory of the application's existence [4].

A later infostealer infection at Context.ai harvested stored OAuth tokens from its user base. One of those tokens belonged to the Vercel employee. That single persistent grant provided access to Vercel's internal dashboards, employee records, API keys, npm tokens, and GitHub credentials [4]. The breach

propagated not because Vercel's perimeter failed in the conventional sense, but because an OAuth grant from a shadow AI application had been silently resident in their identity layer for months.

The Vercel incident fits a pattern documented more broadly by Push Security: on average, organizations have seventeen unique AI application integrations per tenant in Microsoft and Google environments, most of which were authorized by individual employees without IT visibility and most of which have never been reviewed or revoked [4]. The 2025 Salesloft and Gainsight campaign—attributed in [4] to a group Push Security refers to as "Scattered Lapsus$ Hunters," a non-standard designation not found in mainstream threat intelligence taxonomies—demonstrated the downstream reach of this exposure, with OAuth tokens from those vendors enabling lateral movement into more than 1,000 downstream Salesforce and Google Workspace tenants, including multiple major technology and security vendors [4].

Recommendations

Immediate Actions

Security and identity teams should treat OAuth grant management as an active incident surface, not a periodic hygiene task. An immediate audit of all OAuth applications authorized within Microsoft and Google tenants is warranted, with particular attention to applications that have not been used in more than 90 days, applications with broad delegated permissions (Mail.ReadWrite, Files.ReadWrite.All, Calendars.ReadWrite), and any application that is not recognized as an IT-sanctioned deployment. Revocation of stale or shadow grants eliminates the persistent access that post-exploitation and supply chain pivots depend on.

Microsoft 365 administrators should evaluate whether the OAuth device code authorization flow is required across the organization. In environments where it can be disabled entirely through Conditional Access—restricting the flow to explicitly registered, compliant devices—the EvilTokens device code phishing vector is eliminated. Note that this control does not address consent phishing through the authorization code grant flow, which requires the complementary controls described below. Where device code flows must be permitted for specific user populations or device types, Conditional Access policies should enforce compliant device enrollment as a precondition.

Development and security operations teams should audit AI developer tooling credentials with the same rigor applied to cloud IAM service accounts. OAuth tokens used by AI development environments, API keys used by agentic pipelines, and MCP server configurations should be inventoried, scope-limited to

the minimum necessary permissions, and included in credential rotation schedules. Lifecycle hooks in npm packages and similar package manager mechanisms should be treated as a code execution surface, not an administrative footnote.

Short-Term Mitigations

Organizations should deploy or enable Continuous Access Evaluation (CAE) where supported by their identity provider. CAE propagates token revocation signals within seconds to minutes, reducing the default one-hour persistence window to near-real-time for supported resource providers—a substantial improvement over standard token expiration windows. Paired with risky sign-in signals from Microsoft Entra ID Protection or equivalent tooling, CAE substantially narrows the window during which a stolen token remains usable.

Token binding, where the identity provider ties refresh tokens to specific device credentials, prevents token replay from attacker infrastructure even when a token has been successfully exfiltrated. Microsoft's Continuous Access Evaluation extension and Google's Token Binding implementation support this capability for managed device populations. Extending managed device enrollment to developer workstations—a population often exempted from management policies to preserve flexibility—is a high-priority mitigation given the demonstrated targeting of AI developer credentials.

Application allow-listing for OAuth integrations, enforced through Entra ID's application consent policies or equivalent Google Workspace mechanisms, prevents employees from authorizing unreviewed third-party applications. Configuring tenant policies to require administrator approval for applications requesting sensitive delegated permissions eliminates the shadow grant pathway that the Vercel breach exploited.

Strategic Considerations

The structural vulnerability that consent phishing exploits—the cognitive mismatch between the user's perceived security action (the password, the MFA confirmation) and the actual security-critical action (the OAuth consent click)—is unlikely to be fully resolved through incremental controls alone. Long-term resilience requires rethinking how organizations manage the identity of non-human actors, including AI agents, automated pipelines, and developer tooling.

Workload identity federation and hardware-bound credential standards should be evaluated for all AI agent deployments with access to sensitive data or privileged APIs. These approaches tie credential validity to cryptographic attestation of the originating workload's identity, eliminating the portability that

makes stolen refresh tokens valuable to attackers. CSA's "Identity and Access Gaps in the Age of Autonomous AI" [9] provides a framework for this architectural shift, addressing organizational readiness gaps in AI agent identity management that consent phishing campaigns directly exploit.

Organizations should also build ongoing visibility into OAuth grant state as a security capability, not a one-time audit output. Tools that continuously inventory authorized applications, surface anomalous permission grants, and alert on new authorizations outside approved application catalogs provide the detection layer that OAuth's design—optimized for user convenience, not security oversight—does not natively provide.

CSA Resource Alignment

This incident pattern maps directly to several layers of the MAESTRO Agentic AI Threat Modeling Framework [8]. Layer 3 (Agent Frameworks) captures the risk of OAuth tokens and API keys stored in agent configuration files and credential stores, which represent persistent credential exposures when agents process adversarial inputs or when host systems are compromised. Layer 6 (Security and Compliance) addresses the identity governance obligations that OAuth grant management implicates—the authorization scopes, consent grants, and lifecycle controls through which agents act on users' behalf become attack surfaces when those grants are not actively managed.

The CSA AI Controls Matrix (AICM) [11] addresses the identity and access management obligations most directly implicated by this threat class, including controls on privileged access management, credential hygiene, and third-party application governance for AI deployments. The CSA publication "Identity and Access Gaps in the Age of Autonomous AI" [9] documents organizational readiness gaps in AI agent identity management that are directly exploited by consent phishing campaigns targeting agentic workloads.

CSA's Zero Trust guidance and the "Using Zero Trust to Counter Identity Spoofing and Abuse" white paper [10] speak to the device binding and continuous verification principles that would constrain token replay attacks. The principle that no credential should be trusted based solely on its possession—without corroborating device posture, location, and behavioral context—is the architectural foundation that consent phishing consistently undermines and that Zero Trust implementation is designed to restore.

References

[1] Sekoia Threat Detection & Research. "New widespread EvilTokens kit: device code phishing as-a-service - Part 1." Sekoia Blog, February 2026.

[2] BleepingComputer. "ConsentFix v3 attacks target Azure with automated OAuth abuse." BleepingComputer, May 2026.

[3] Mitiga. "MCP Token Theft in Claude Code: A Man-in-the-Middle Attack." Mitiga Research, April 2026.

[4] Push Security. "Unpacking the Vercel breach: Shadow AI and OAuth sprawl." Push Security Blog, April/May 2026.

[5] Cloud Security Alliance AI Safety Initiative. "OAuth Device Code Phishing Hits 340+ Microsoft 365 Organizations." CSA Labs, March 25, 2026.

[6] Proofpoint. "Device Code Phishing is an Evolution in Identity Takeover." Proofpoint Threat Insight, 2026.

[7] Beyond Identity. "The Attacker Gave Claude Their API Key: Why AI Agents Need Hardware-Bound Identity." Beyond Identity, 2026.

[8] Cloud Security Alliance. "Agentic AI Threat Modeling Framework: MAESTRO." CSA Blog, February 6, 2025.

[9] Cloud Security Alliance. "Identity and Access Gaps in the Age of Autonomous AI." CSA Research, March 2026.

[10] Cloud Security Alliance. "Using Zero Trust to Counter Identity Spoofing and Abuse." CSA Research, 2026.

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

