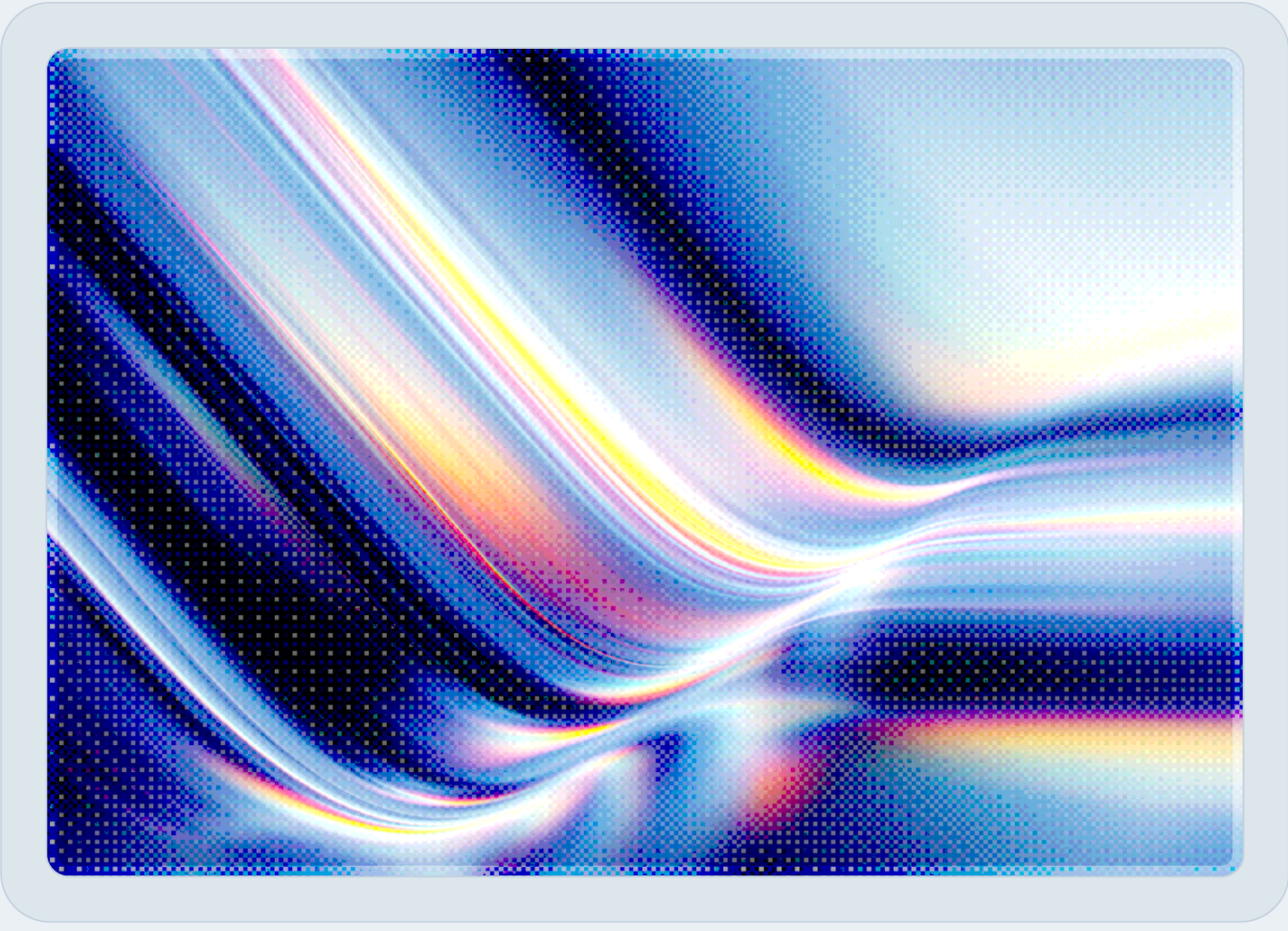
CSAI Foundation | Cloud Security Alliance

EvilTokens: Device-Code Phishing Renders MFA Irrelevant

OAuth Device Authorization Abuse and Its Implications for AI Agent Deployments

2026-05-20

AI-assisted Rapid Research



CSAI | CSA cloud security alliance®

Key Takeaways

- EvilTokens is a Phishing-as-a-Service (PhaaS) platform launched in mid-February 2026 that weaponizes the OAuth 2.0 Device Authorization Grant to steal persistent Microsoft 365 access tokens—without capturing any credentials and without triggering any MFA challenge the victim could recognize as suspicious.
- The attack does not defeat MFA; it redirects it. Victims complete genuine, MFA-verified authentication on legitimate Microsoft infrastructure, but the resulting tokens are issued to the attacker's OAuth client rather than the victim's device, making conventional MFA controls structurally irrelevant to the outcome.
- Refresh tokens harvested through device-code phishing carry a 90-day rolling validity window and survive password resets; organizations that reset credentials without explicitly revoking outstanding tokens have not remediated the compromise [1].
- EvilTokens integrates large language models—Meta's Llama 3.1 and 3.3 series and OpenAI's GPT-4o mini—to automate post-compromise email triage and business email compromise (BEC) scenario generation, pairing token theft with AI-driven monetization at machine speed [2].
- Push Security documented a 37.5-fold increase in detected device-code phishing pages in 2026, up from a 15-fold increase measured at the start of March, driven primarily by EvilTokens campaign activity [7].
- AI agents that hold delegated OAuth tokens for Microsoft 365 and Graph API access are directly exposed: a single compromised agent token store can yield persistent access to every resource the agent is authorized to reach, with no further interaction with the authorizing user.

Background

The OAuth Device Authorization Grant

The OAuth 2.0 Device Authorization Grant, standardized in IETF RFC 8628, was designed to solve a narrow, practical problem: enabling applications running on input-constrained hardware—smart televisions, command-line tools, IoT terminals—to obtain user authorization when the device cannot

render a browser-based login interface [11]. In the intended flow, the device requests a short-lived code from the authorization server and instructs the user to visit a verification URL on a secondary device, sign in there, and enter the code. The authorization server polls for completion and returns tokens to the originating device once the user approves.

The convenience that makes this flow valuable for constrained environments creates an exploitable trust gap. The authorization step and the authentication decision are deliberately decoupled across two devices, and the flow does not require the user to explicitly identify the application being authorized—only to enter a code at a legitimate login page. An attacker who can substitute their own device code into this workflow places themselves in the position of the application receiving authorization, with the user providing proof of identity and completing MFA against infrastructure they correctly recognize as legitimate.

EvilTokens PhaaS

EvilTokens is a Phishing-as-a-Service platform first documented by Sekoia's Threat Detection and Research team on March 30, 2026, with phishing pages observed in the wild starting in mid-February 2026 [1]. The platform was marketed via Telegram with a commercial model that includes a $1,500 one-time fee plus $500 monthly subscription for the Office 365 device-code capture kit, supplemented by $600 for a B2B sender module, $1,000 for an SMTP sender, and a $500 lifetime license for a multi-account portal tool [2]. Sekoia documented approximately 280 subscribers in the EvilTokens private Telegram channel as of March 19, 2026, with more than 1,000 domains hosting EvilTokens phishing pages identified by March 23 [1][2]. A campaign tracked through March 2026 compromised more than 340 Microsoft 365 organizations across the United States, Canada, France, Australia, India, Switzerland, and the UAE, spanning financial services, healthcare, government, construction, manufacturing, legal, and nonprofit sectors [3][8].

The technical mechanism is straightforward to execute at scale: the kit generates device codes dynamically, deploys phishing lures impersonating Adobe Acrobat, DocuSign, voicemail notifications, and other familiar business documents, and directs victims to enter the attacker-generated code at the legitimate Microsoft verification portal. No fake login page is involved. Victims interact exclusively with authentic Microsoft infrastructure.

What distinguishes EvilTokens from earlier device-code phishing scripts is the depth of its post-compromise automation. The platform integrates Groq's Llama 3.1 (8B parameter) model to ingest up to 5,000 harvested emails and identify exploitable payment threads for financial fraud targeting. It uses Llama 3.3 (70B parameter) to generate contextually appropriate BEC attack messages tailored to each victim's role and business relationships, and OpenAI's GPT-4o mini to automatically translate stolen email content into English, enabling operators to target organizations regardless of language [2].

BleepingComputer has reported that EvilTokens's developers have publicly announced plans to extend the kit to support Gmail and Okta environments, indicating active development with scope beyond the current Microsoft ecosystem [8].

Security Analysis

Why MFA Provides No Defense

A common misapplication of MFA-as-defense to device-code phishing conflates authentication strength with authorization scope—the attack does not require defeating MFA; it redirects the authorization output. The attack does not involve a stolen password, an intercepted authentication challenge, or a fake login page. It changes what MFA is authorizing, not whether MFA fires.

When a victim enters an attacker-generated device code at `microsoft.com/devicelogin` and completes push notification, TOTP, or SMS verification, they have successfully authenticated their identity against Microsoft's infrastructure. What they have not controlled is which OAuth client receives the resulting authorization. That client is the attacker's registered application. The access token and refresh token produced by the authentication event are returned to the attacker's backend, not to the victim's device. A security operations center monitoring for MFA fatigue attacks, impossible travel, or suspicious sign-in geography may observe nothing anomalous: the sign-in occurred from the victim's own device, on a domain the victim correctly recognizes as legitimate, with MFA successfully completed.

Microsoft's April 2026 analysis of an active EvilTokens campaign documented a technical refinement that further compresses the detection window: operators dynamically generated device codes at the moment of victim interaction, ensuring codes remained within their standard 15-minute validity window regardless of any delay between phishing email delivery and victim action [4]. Campaign infrastructure was deployed on Railway.com's platform-as-a-service environment to manage concurrent authorization polling across thousands of simultaneous sessions, enabling multiple independent campaigns to operate in parallel.

Token Persistence and Post-Compromise Dwell

Access tokens issued through device-code flow carry a 60-to-90-minute validity window by default, though tenant administrators can modify these defaults through token lifetime policies [13]. Refresh tokens carry a 90-day rolling window: each time the attacker uses the refresh token to request a new access token, the 90-day clock resets [1]. This creates a persistence mechanism that is structurally

independent of the victim's credentials. A user whose password is reset following a phishing incident retains an outstanding attacker-held refresh token that remains valid for its full window unless explicitly revoked.

The practical consequence is that the most common first-line incident response action—credential reset—does not constitute remediation for device-code phishing compromise. Microsoft's Entra platform provides a `revokeSignInSessions` API endpoint that invalidates all active refresh tokens for an account; calling this endpoint is a necessary component of any remediation sequence, and its absence from standard credential-reset procedures likely represents a structural gap in incident response workflows at many organizations that have not specifically updated playbooks for token-based credential theft [4][5]. Organizations that have not mapped token revocation into their identity incident response playbooks will systematically underestimate the dwell time of device-code phishing compromises, treating password reset as closure when attacker access continues.

EvilTokens's Outlook-impersonating webmail interface exploits this persistence by maintaining a continuously refreshed session that allows operators to read, search, and send email on behalf of the victim, enumerate SharePoint and OneDrive files, access Teams conversations, and conduct reconnaissance through the Microsoft Graph API—all without any further interaction with the victim and without any new authentication event appearing in identity provider logs [2].

AI Agent Exposure

AI agents running in headless contexts commonly rely on the device authorization grant or similar non-interactive flows—the IETF standard explicitly targets this use case [11], and many enterprise integrations with Microsoft 365 and Graph API are built on this assumption. Agents authorized to act on behalf of human users against SharePoint, Teams, or other productivity environments hold delegated OAuth credentials—including refresh tokens—that constitute their operational identity for all subsequent access.

An agent's token store becomes a high-value target in two structurally distinct ways. In the first scenario, an attacker targets the enrollment workflow itself: if an administrator conducting agent setup can be socially engineered into entering an attacker-generated device code during the authorization flow, the attacker captures the agent's refresh token at issuance and inherits its full delegated permissions from that moment forward. In the second scenario, the attacker accesses the agent's runtime token store directly, through container filesystem access, environment variable extraction, secrets manager misconfiguration, or lateral movement from a compromised deployment pipeline.

In enterprise productivity automation deployments where agents operate with tenant-wide Graph API scopes, the blast radius of agent token theft can substantially exceed that of a single user account compromise, because a single agent token may authorize access across all users and data types in the

agent's scope. Agents built for enterprise productivity automation may request broad Graph API scopes – `Mail.ReadWrite`, `Files.ReadWrite.All`, `Calendars.ReadWrite`, `Sites.ReadWrite.All`, and `User.ReadBasic.All` –when they must operate autonomously across multiple data types. A single compromised agent token provides access to all of these surfaces simultaneously, for whichever users' data the agent is authorized to process. CSA's March 2026 research on AI agent identity management documents the maturity gap in organizational controls for non-human identity and the challenges enterprises face in detecting and attributing agent-sourced activity during incident response—a gap that directly extends the dwell time of agent token compromises relative to equivalent human account incidents [6].

The AI-augmented capabilities that EvilTokens brings to post-compromise operations are a direct mirror of what enterprise AI agents do legitimately. Where human threat actors reviewing stolen sessions are rate-limited by attention, the platform's LLM pipeline can process thousands of messages, identify financial exposure, draft contextually appropriate BEC emails, and stage follow-on BEC operations against the victim's contacts without the attention or time constraints that limit manual threat actor operations. An organization that has deployed AI agents to automate email handling across executive accounts has inadvertently created high-value targets that, if compromised, can be weaponized through the same AI automation capabilities the organization itself relies upon.

Threat Actor Landscape

Device-code phishing as a tradecraft capability predates EvilTokens's commercial availability by at least 18 months. Storm-2372, a threat actor Microsoft assesses with moderate confidence as aligned with Russian state interests, conducted device-code phishing campaigns against government, NGO, defense, telecommunications, healthcare, and energy sector targets across Europe, North America, Africa, and the Middle East from at least August 2024 through the date of Microsoft's February 2025 disclosure [5]. Storm-2372's operational approach combined social engineering through Microsoft Teams, WhatsApp, and Signal—with operators impersonating prominent individuals to build relationship context—before delivering device-code lures to establish a more credible authorization pretext.

The commercial availability of EvilTokens has broadened this tradecraft from nation-state and sophisticated eCrime operators to a substantially wider threat actor population. Push Security measured a 15-fold increase in detected device-code phishing pages targeting Microsoft 365 tenants at the start of March 2026, a figure that had risen to 37.5-fold by the time of their published analysis, with EvilTokens identified as the most prominent of multiple tracked kits and campaigns [7]. Tycoon 2FA, a competing adversary-in-the-middle PhaaS operation whose infrastructure was seized in a coalition action led by Microsoft and Europol in March 2026, resumed device-code phishing operations by April 2026 after restoring from backed-up codebases with matching encryption keys and backend patterns,

demonstrating the resilience of this technique to even coordinated infrastructure disruption [12]. The convergence of nation-state tradecraft, organized eCrime adoption, and now multiple commercial PhaaS offerings marks device-code phishing as a likely enduring capability in the threat landscape—one whose resilience has already been demonstrated against infrastructure disruption—rather than a technique that a single takedown action can address.

Recommendations

Immediate Actions

The most direct technical control against device-code phishing is a Conditional Access policy that explicitly blocks the Device Code Flow authentication condition for users and workloads that have no documented operational requirement for it—where operationally feasible, this is the recommended first action [4][5]. In Microsoft Entra ID, this policy should target all users and all cloud applications, apply the Authentication Flows condition scoped to Device Code Flow, and set the grant control to Block. Administrators should enumerate and explicitly exclude any legitimate applications that require the flow—approved meeting-room hardware, CI/CD pipeline tools, managed IoT deployments—before enforcing the policy broadly. The exclusion list should be treated as a standing security asset, reviewed periodically, and protected against unauthorized additions.

Organizations with any reason to believe they have been targeted should call `revokeSignInSessions` for affected accounts immediately and not treat password reset as an equivalent or sufficient action. Entra sign-in logs should be queried for authentication events showing Device Code Flow as the client authentication type and cross-referenced against the documented exclusion list to identify any unauthorized token issuance. Inbox rule modifications, external email forwarding configuration, and SharePoint access events in the days following any identified device-code authentication should be reviewed as indicators of post-compromise activity.

For AI agent deployments, token stores should be audited to identify any credentials issued through device-code authorization. Any agent whose enrollment workflow cannot be confirmed as having used an authorized device code should be treated as potentially compromised and its refresh token revoked and reissued through a controlled re-enrollment process.

Short-Term Mitigations

Where device-code flow cannot be fully blocked due to legitimate operational dependencies, network-layer constraints provide a meaningful secondary control. Microsoft Entra Conditional Access and Okta's application-level network policy both support restricting which IP address ranges may successfully complete device-code authorization for a given application [9]. Limiting device-code polling to documented corporate IP ranges or VPN egress addresses substantially increases the operational cost for remote threat actors who cannot control the source address of their authorization polling requests.

AI agent OAuth configurations should be reviewed for scope breadth. Agents should operate under strict least-privilege access, holding only the specific Graph API permissions their documented function requires. Token stores should be protected using enterprise secrets management platforms—Azure Key Vault, AWS Secrets Manager, HashiCorp Vault—rather than environment variables or container filesystem locations accessible to a broad set of infrastructure principals. Explicit token rotation schedules and revocation procedures should be written into agent operational runbooks rather than treated as implicit follow-ons to human identity incident response processes.

Strategic Considerations

Phishing-resistant authentication factors provide structural protection against device-code phishing in a way that conventional MFA does not. FIDO2 hardware security keys and passkeys using platform authenticators bind the cryptographic authentication assertion to the originating relying party. Where Conditional Access policies combine phishing-resistant authentication requirements with device-compliance or network-based controls, they can restrict the attacker's backend from successfully completing the authorization polling step, because the policy evaluates session characteristics at token issuance rather than solely at the authentication event. The protection is therefore a property of how Entra evaluates the full authorization context, not of the authenticator detecting a relying-party mismatch—device-code flow routes authentication through the legitimate Microsoft domain, so no RP mismatch exists for the authenticator to observe. Organizations should prioritize phishing-resistant factor adoption for high-risk populations—executives, system administrators, and anyone who owns or oversees AI agent deployments—and configure Conditional Access policies in Entra, Okta, and Google Workspace to require phishing-resistant authentication for privileged and high-value contexts [5][9].

The deeper strategic issue device-code phishing surfaces is non-human identity governance. AI agents, service accounts, automation pipelines, and API integrations collectively hold a large and growing share of the OAuth tokens in active use in most enterprise environments, yet the identity and access management programs most organizations operate were designed primarily around human identity lifecycle processes—hire, role change, departure. The credential rotation, scope review, and revocation

workflows that govern human accounts are frequently absent for agent credentials, creating a class of long-lived, broadly scoped OAuth tokens with no automatic expiration, no periodic access review, and no clear incident response procedure tied to them. CSA's research on autonomous AI identity management provides a baseline assessment of how enterprises currently handle this gap and identifies the governance investments required to close it [6].

CSA Resource Alignment

CSA's MAESTRO framework for agentic AI threat modeling engages multiple layers of the attack surface this note describes. Layer 3 (Agent Trust Boundaries) addresses the conditions under which AI agents are authorized to act on behalf of human principals and inherit their delegated permissions—the precise trust relationship that device-code phishing exploits both during agent enrollment workflows and through post-compromise token abuse. Layer 5 (Infrastructure and Deployment Security) covers the runtime environments and secrets management configurations where agent OAuth tokens are held and must be protected. The post-compromise data exfiltration and BEC operations enabled by EvilTokens's AI-augmented analysis pipeline engage Layer 6 (Data and Model Inputs), particularly the risks that adversarial access to agent-accessible data creates for downstream data integrity and organizational trust.

CSA's AI Controls Matrix (AICM) provides control-level guidance directly applicable to the identity hardening measures this note recommends. Controls governing non-human identity lifecycle management, credential rotation policy, and access scope governance for AI agents map to the specific agent token hardening actions described in the Recommendations section. Organizations implementing AICM controls for AI agent identity will find that the device-code phishing scenario provides a concrete forcing function for making those controls operational rather than aspirational.

Two recent CSA publications provide foundational context for organizations operationalizing a response. "Identity and Access Gaps in the Age of Autonomous AI" (March 2026) documents the maturity gap in enterprise controls for AI agent credentials, surveys how organizations currently manage non-human identity at scale, and identifies the governance investment required to reduce agent credential exposure [6]. "Using Zero Trust to Counter Identity Spoofing and Abuse" (April 2026) addresses phishing-resistant authentication design and zero trust identity controls applicable to both human and non-human identity contexts [10]. Read together, these publications give security teams the framework and vocabulary to address device-code phishing not as an isolated technical incident but as a symptom of a broader identity governance gap that agentic AI deployments have made urgent.

References

[1] Sekoia Threat Detection & Research. "New widespread EvilTokens kit: device code phishing as-a-service – Part 1." Sekoia Blog, March 30, 2026.

[2] Sekoia Threat Detection & Research. "EvilTokens: an AI-augmented Phishing-as-a-Service for automating BEC fraud – Part 2." Sekoia Blog, April 7, 2026.

[3] The Hacker News. "Device Code Phishing Hits 340+ Microsoft 365 Orgs Across Five Countries via OAuth Abuse." The Hacker News, March 2026.

[4] Microsoft Security Blog. "Inside an AI-enabled device code phishing campaign." Microsoft, April 6, 2026.

[5] Microsoft Security Blog. "Storm-2372 conducts device code phishing campaign." Microsoft, February 13, 2025.

[6] Cloud Security Alliance. "Identity and Access Gaps in the Age of Autonomous AI." CSA, March 23, 2026.

[7] Push Security. "Analyzing the rise in device code phishing attacks in 2026." Push Security, April 4, 2026.

[8] BleepingComputer. "New EvilTokens service fuels Microsoft device code phishing attacks." BleepingComputer, April 1, 2026.

[9] Okta. "Device code phishing: it's phishing with dynamite." Okta, May 11, 2026.

[10] Cloud Security Alliance. "Using Zero Trust to Counter Identity Spoofing and Abuse." CSA, April 22, 2026.

[11] IETF. "RFC 8628 – OAuth 2.0 Device Authorization Grant." IETF, August 2019.

[12] eSentire Threat Response Unit. "Tycoon 2FA Operators Adopt OAuth Device Code Phishing." eSentire, April–May 2026.

[13] Microsoft. "Configure token lifetime policies in Microsoft Entra ID." Microsoft Learn, 2024.

