OAuth Device Code Phishing Hits 340+ Microsoft 365 Organizations

How Attackers Weaponize a Legitimate Protocol Flow to Bypass MFA at Scale

Unofficial AI-assisted Research

Cloud Security Alliance AI Safety Initiative

2026-03-25

Key Takeaways

- OAuth device code phishing weaponizes a legitimate protocol feature (RFC 8628) to harvest Microsoft 365 access tokens without requiring victims to enter credentials on a lookalike site, making the authentication interaction indistinguishable from genuine sign-in for the end user — though behavioral anomalies in sign-in logs may be detectable by security teams.
- Multifactor authentication provides no protection against this attack class: the victim completes the MFA challenge themselves on behalf of the attacker, and the resulting refresh tokens persist even after a password reset, complicating remediation.
- What began as a state-sponsored Russian intelligence technique in mid-2024 has commoditized into a Phishing-as-a-Service (PhaaS) offering; the EvilTokens platform, launched in February 2026, enabled a campaign that compromised 340+ organizations across five countries within weeks [1][2].
- Microsoft Entra ID Conditional Access policies can block device code flow entirely and represent the most direct technical control available to defenders [3].
- Organizations should audit all legitimate uses of device code flow before blocking it and supplement that control with token revocation procedures, anomaly monitoring on Entra sign-in logs, and network-level blocking of identified attacker infrastructure.

Background

The OAuth 2.0 Device Authorization Grant, defined in RFC 8628 [11], was designed to allow input-constrained devices — smart televisions, printers, IoT sensors — to obtain delegated access on behalf of a user without requiring those devices to display a browser or accept keyboard input. The protocol works through a deliberate separation of channels: the device obtains a short-lived code pair from the authorization server and instructs the user to complete authentication on a separate, capable device by visiting a well-known URL and entering a human-readable code. Once the user authenticates and approves, the constrained device exchanges its code for access and refresh tokens.

Microsoft supports this flow for Microsoft 365 and Azure resources via the `login.microsoftonline.com/oauth2/v2.0/devicecode` endpoint, and the legitimate verification URL is `microsoft.com/devicelogin`. This is an entirely intentional feature of the platform with valid operational uses in enterprise environments, which is precisely what makes it attractive as an attack vector. Because there is no lookalike domain, no credential-harvesting form, and no suspicious redirect — only the real Microsoft authentication page — traditional phishing detection mechanisms offer limited protection. Security awareness training that focuses on URL inspection or certificate verification does not address this attack class.

The attack's origins in public threat intelligence trace to mid-2024, when Microsoft began tracking a cluster it designated Storm-2372. Microsoft publicly disclosed the campaign in February 2025, attributing it with high confidence to a Russia-aligned threat actor targeting government agencies, nongovernmental organizations, defense contractors, telecommunications providers, and energy sector entities across Europe, North America, Africa, and the Middle East [4]. Concurrent with Microsoft's disclosure, the threat intelligence firm Volexity published findings linking at least two additional Russia-affiliated clusters — tracked as UTA0304 and UTA0307 — to the same technique, suggesting independent parallel development or knowledge-sharing within the Russian intelligence community [5].

By the close of 2025, the technique had spread beyond documented state-sponsored campaigns. Proofpoint documented a sharp volumetric increase in device code phishing activity beginning in September 2025 and identified a financially motivated threat actor, designated TA2723, adopting the method by October 2025 [6]. This broader adoption followed a pattern observed in other advanced techniques: as operational playbooks became publicly documented through threat research, actors with lower technical barriers incorporated the method into off-the-shelf tooling. The emergence of the EvilTokens Phishing-as-a-Service platform in February 2026 marked the full commoditization of this technique into an accessible service model.

Security Analysis

Attack Mechanics

The attacker initiates the attack by sending a POST request to Microsoft's device authorization endpoint with a chosen `client_id`, receiving in response a `user_code` (a short alphanumeric string), a `device_code`, a `verification_uri`, and an expiration window — typically fifteen minutes. The attacker then crafts a phishing message directing the victim to `microsoft.com/devicelogin` and presenting the `user_code` as if it were a one-time password, an access code for a shared

document, or a security verification token. The victim, interacting entirely with legitimate Microsoft infrastructure, enters the code and completes whatever MFA challenge their tenant enforces. At this moment the attacker's polling loop against the `/token` endpoint succeeds, and Microsoft returns a valid `access_token` and `refresh_token` to the attacker's system.

The refresh token is the more consequential artifact. Access tokens are short-lived, but refresh tokens can remain valid for weeks or months depending on tenant configuration, and they survive password resets. An organization that discovers a compromise and resets the affected user's password without also revoking the token session may unknowingly leave the attacker's access intact. Token revocation requires an explicit call to the `revokeSignInSessions` API or an equivalent administrative action in Microsoft Entra ID [3][12].

After obtaining tokens, observed post-compromise activity has consistently followed a pattern of Microsoft Graph API enumeration: accessing mailbox contents, downloading OneDrive files, harvesting contacts and calendar data, and in some cases registering attacker-controlled devices against the victim's Entra ID tenant to establish persistent footholds that survive token revocation [5].

The EvilTokens Campaign and Scale

The EvilTokens PhaaS platform launched publicly on Telegram on February 16, 2026 [1]. The platform offered three service tiers — broadly corresponding to email delivery, token capture, and SMTP relay capabilities — and was advertised as incorporating AI-assisted features to tailor lure content and improve deliverability against enterprise email filtering. Infrastructure analysis published by Huntress in March 2026 identified the platform's backend as hosted on Railway.com, a legitimate Platform-as-a-Service provider, with Cloudflare Workers and Vercel intermediaries used to obfuscate redirect chains [2].

The campaign attributable to EvilTokens infrastructure, first observed on February 19, 2026, had reached 340 or more Microsoft 365 organizations across the United States, Canada, Australia, New Zealand, and Germany by mid-March [1]. Affected sectors included construction, nonprofit organizations, real estate, manufacturing, financial services, healthcare, legal services, and local government. The breadth of sector targeting reflects the indiscriminate, opportunistic nature of PhaaS operations, which prioritize volume over selectivity.

Lure variants in the campaign impersonated construction bid solicitations, DocuSign document sharing notifications, voicemail delivery alerts, Microsoft Forms prompts, and shared file reminders incorporating victim organization branding derived from publicly available sources. A consistent anti-analysis pattern was observed on landing pages: right-click was disabled, developer tools were blocked, and JavaScript debugger loops were introduced to complicate automated analysis [1].

Attacker Infrastructure

Huntress researchers identified five Railway.com IP addresses responsible for the majority of observed authentication polling traffic in the EvilTokens campaign [2]. Three of those five addresses accounted for approximately 84% of events, suggesting centralized infrastructure rather than a distributed or ephemeral architecture. This concentration creates an actionable blocking opportunity, though defenders should expect infrastructure rotation as the campaign matures.

EvilTokens landing pages exhibited two distinctive HTTP indicators: an API endpoint pattern using `/api/device/start` and `/api/device/status/` paths, and a non-standard HTTP header — `X-Antibot-Token` — inserted by the platform's backend [2]. These signatures may facilitate detection by organizations with full packet inspection or cloud access security broker (CASB) visibility.

State-sponsored actors in earlier campaign waves used domain infrastructure including `sen-comms[.]com`, `afpi-sec[.]com`, `chromeelevationservice[.]com`, and `comms-net[.]com` for UTA0304, and `connect-71q.pages[.]dev` and `rosejob[.]com` for UTA0307 [5]. A separate cluster designated UTA0352, tracked by Volexity from March 2025, exploited the Microsoft Visual Studio Code OAuth flow rather than the standard device code endpoint, using the `insiders.vscode.dev` redirect URI to make token requests appear to originate from a development tooling integration [5]. This variant demonstrates that the underlying technique — generating OAuth tokens through legitimate Microsoft redirect URIs — is not limited to a single endpoint or `client_id` value.

Detection Opportunities

Microsoft Entra ID sign-in logs provide the primary detection surface. Log entries for device code authentication carry the fields `authenticationProtocol: deviceCode` and `originalTransferMethod: deviceCodeFlow`, which are not present in conventional interactive sign-in events. Organizations that do not use device code flow for any legitimate purpose can treat any occurrence of these fields as a strong indicator warranting immediate investigation. Organizations with legitimate device code uses should baseline the expected `client_id` values, geographic origins, and user populations, and alert on deviations.

Additional behavioral indicators include sign-ins originating from VPS providers, Tor exit nodes, and commercial VPN services; the presence of `python-requests/2.25.1` as a User-Agent string in token polling traffic; and post-authentication Microsoft Graph API activity involving bulk mail access or file downloads in short time windows. Unauthorized device registrations in Microsoft Entra ID —

particularly following a sign-in event where device code authentication was logged — should be treated as a high-priority indicator of potential compromise requiring immediate investigation, particularly when combined with other anomalous signals.

MFA Bypass Implications

This attack class merits particular attention from organizations that have completed MFA deployments and consider themselves well-protected against credential phishing. Device code phishing does not bypass MFA in a technical sense; it causes the victim to complete MFA authentically on the attacker's behalf. The security assumption embedded in most MFA deployments — that possession of the authentication factor correlates with user intent — does not hold when the authentication flow is initiated by an attacker and the user is socially engineered into approving it.

This structural limitation also applies to phishing-resistant authenticators such as passkeys and hardware tokens when used in device code flow contexts. Because authentication occurs on genuine Microsoft infrastructure, FIDO2 origin binding provides no protection here — the authenticator correctly validates the origin. The vulnerability lies in the device code protocol itself, which allows an attacker to initiate a flow and collect the resulting token after the user authenticates. The correct control is to restrict the attack surface by disabling the flow, not to add authentication factors to a flow that remains architecturally exploitable.

Recommendations

Immediate Actions

Organizations should prioritize the following within 72 hours of reading this note, to limit ongoing exposure from active campaigns. First, query Microsoft Entra ID sign-in logs for any historical occurrence of `authenticationProtocol: deviceCode` over the preceding 90 days (aligning with the default Entra ID sign-in log retention period), using the Microsoft Entra admin center or Microsoft Sentinel, and triage any results for unauthorized access patterns. Second, block the five Railway.com IP addresses identified in the EvilTokens campaign (`162.220.234[.]41`, `162.220.234[.]66`, `162.220.232[.]57`, `162.220.232[.]99`, `162.220.232[.]235`) by adding them to Entra Named Locations and excluding them from all access policies, or by blocking them at the network perimeter [2]. Third, review all active Entra ID Conditional Access policies to confirm whether device code flow is currently restricted, and if not, identify a path to enabling that restriction.

For any user account where unauthorized device code authentication is confirmed, administrators should immediately call `revokeSignInSessions` [12] and audit Microsoft Entra ID for any devices registered within 48 hours of the suspicious sign-in event. Unauthorized registered devices should be removed, and affected accounts should be reviewed for mail forwarding rules, delegated access grants, and OAuth application consents added after the compromise date.

Short-Term Mitigations

The highest-impact sustained control is a Conditional Access policy targeting the Authentication Flows condition with the Device Code Flow option selected and the policy Grant action set to Block. Microsoft introduced this policy condition specifically in response to the abuse pattern documented beginning in 2025 [3]. Before enabling the policy in enforcement mode, organizations should audit their environment for any managed devices, service accounts, or operational workflows that depend on device code flow — legacy IoT integrations, shared kiosk configurations, and some print management systems represent typical legitimate use cases to evaluate — and migrate those use cases to alternative authentication mechanisms before blocking.

Organizations should also enforce admin consent requirements for OAuth application registrations, preventing users from independently authorizing third-party applications that request Microsoft Graph API scopes. Conditional Access policies requiring device compliance or Hybrid Azure AD Join status add a meaningful barrier even when token theft occurs, since the attacker's system is unlikely to satisfy device compliance requirements for subsequent access attempts using stolen tokens.

Monitoring for the `X-Antibot-Token` header and the `/api/device/start` endpoint pattern at the network layer, where proxy or CASB visibility permits, may identify phishing infrastructure that has not yet been publicly catalogued. Given the pace of infrastructure rotation among PhaaS operators, behavioral signatures of this type may prove more durable than IP-based blocklists.

Strategic Considerations

The commoditization of device code phishing into a PhaaS model indicates that this attack vector will persist as a feature of the threat landscape for the foreseeable future. Organizations that treat it as a point-in-time incident requiring a one-time response will likely face repeated exposure. The more durable posture is to treat device code flow as a legacy authentication mechanism whose risk profile exceeds its operational value for most enterprise use cases — and to eliminate it from the environment entirely via Conditional Access policy, following the same deprecation logic that organizations have applied to basic authentication and legacy protocols.

Security awareness programs should be updated to include device code phishing as a distinct category. Traditional phishing awareness content that emphasizes URL inspection, sender domain verification, and certificate checking does not address this attack, because every element of the authentication interaction occurs on legitimate Microsoft infrastructure. Users should be trained that any unsolicited code presented as an "OTP," "access token," or "security code" to enter at `microsoft.com/devicelogin` should be treated as a potential attack and reported without entering the code.

At a governance level, organizations should review their OAuth application inventory and establish a process for periodic attestation of authorized application consents and registered device entries in Microsoft Entra ID. Unauthorized OAuth consents and device registrations are frequently left in place long after incident remediation because they fall outside standard user account review workflows.

CSA Resource Alignment

This campaign directly implicates several areas of CSA's guidance portfolio on identity security and cloud access control.

CSA's *Zero Trust Principles and Guidance for IAM* establishes that identity verification must be continuous and context-aware, not one-time at session initiation [7]. Device code phishing exploits the gap between a single verified authentication event and the persistent access it grants; Zero Trust token lifetime policies and continuous access evaluation — both available in Microsoft Entra ID — directly address this gap by re-evaluating access conditions on an ongoing basis rather than treating token issuance as a permanent grant.

The *AI Controls Matrix (AICM)* [10] and its predecessor, the *Cloud Controls Matrix (CCM)*, address identity and access management through the IAM control domain. CCM IAM-02 (Strong Authentication) and IAM-03 (Identifier Management) are relevant to the requirement for strong controls over OAuth authorization flows and token issuance. IAM-09 (User Access Reviews) applies to the periodic attestation of authorized application consents and device registrations that device code phishing can silently introduce into a tenant, ensuring these additions are identified and removed as part of regular access governance [8][10].

CSA's *Confronting Shadow Access: Risks, Considerations for Zero Trust and Artificial Intelligence Deployments* is directly applicable: device code flow creates an access pathway that is invisible to conventional session monitoring and approval workflows, meeting the definition of shadow access when

used by an attacker [9]. The document's recommendations for continuous authorization and session binding apply here.

For organizations using agentic AI workflows that rely on OAuth tokens for Microsoft 365 integration, CSA's *MAESTRO* framework (Agentic AI Threat Modeling) identifies token reuse and session persistence as a threat class for AI agents operating with delegated credentials. Device code-issued tokens used in AI pipeline integrations represent a compound risk: if an attacker obtains such a token, they gain access not only to the user's mailbox and files but potentially to any agentic workflow the token authorizes.

Finally, CSA's *AI Organizational Responsibilities* series – particularly the modules on Governance, Risk Management, and Compliance – frames the organizational accountability dimension: the decision to leave device code flow enabled without compensating controls represents an accepted risk that must be documented, owned, and periodically reassessed as the threat landscape evolves.

References

[1] The Hacker News, "Device Code Phishing Hits 340+ Microsoft 365 Orgs Across Five Countries via OAuth Abuse," March 2026. https://thehackernews.com/2026/03/device-code-phishing-hits-340-microsoft.html

[2] Huntress, "Riding the Rails: Threat Actors Abuse Railway.com PaaS as Microsoft 365 Token Attack Infrastructure," March 20, 2026. https://www.huntress.com/blog/railway-paas-m365-token-replay-campaign

[3] Microsoft Security Blog, "Defending against evolving identity attack techniques," May 29, 2025. https://www.microsoft.com/en-us/security/blog/2025/05/29/defending-against-evolving-identity-attack-techniques/

[4] Microsoft Security Blog, "Storm-2372 conducts device code phishing campaign," February 13, 2025. https://www.microsoft.com/en-us/security/blog/2025/02/13/storm-2372-conducts-device-code-phishing-campaign/

[5] Volexity, "Multiple Russian Threat Actors Targeting Microsoft Device Code Authentication," February 13, 2025; and "Phishing for Codes: Russian Threat Actors Target Microsoft 365 OAuth Workflows," April 22, 2025. https://www.volexity.com/blog/2025/02/13/multiple-russian-threat-actors-targeting-microsoft-device-code-authentication/ ; https://www.volexity.com/blog/2025/04/22/phishing-for-codes-russian-threat-actors-target-microsoft-365-oauth-workflows/

[6] Proofpoint, "Access granted: phishing with device code authorization for account takeover," December 18, 2025. https://www.proofpoint.com/us/blog/threat-insight/access-granted-phishing-device-code-authorization-account-takeover

[7] Cloud Security Alliance, "Zero Trust Principles and Guidance for IAM," CSA Research. https://cloudsecurityalliance.org/research/working-groups/zero-trust

[8] Cloud Security Alliance, "Cloud Controls Matrix v4.0," CSA Research. https://cloudsecurityalliance.org/research/cloud-controls-matrix/

[9] Cloud Security Alliance, "Confronting Shadow Access: Risks, Considerations for Zero Trust and Artificial Intelligence Deployments," CSA Research. https://cloudsecurityalliance.org/research/artifacts/confronting-shadow-access (URL confirmed unavailable at time of publication; refer to the CSA research portal for current location.)

[10] Cloud Security Alliance, "AI Controls Matrix (AICM)," CSA AI Safety Initiative. https://cloudsecurityalliance.org/research/working-groups/ai-safety-initiative (Primary AICM working group page; artifact URL confirmed unavailable at time of publication.)

[11] IETF RFC 8628, "OAuth 2.0 Device Authorization Grant," August 2019. https://www.rfc-editor.org/rfc/rfc8628

[12] Microsoft, "revokeSignInSessions API — Microsoft Graph API Reference." https://learn.microsoft.com/en-us/graph/api/user-revokesigninsessions