INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-03 06:52 UTC

Microsoft's MDASH Agentic Scanner Enters Enterprise Preview: What Security Teams Need to Evaluate Now

SECURITY ANALYSIS | MEDIUM | CVSS 5.0

SCC Item ID	SCC-STY-2026-0164
Type	Security Analysis
Severity	MEDIUM
CVSS Base Score	5.0
Affected Products	Microsoft Defender, GitHub Code Security (GitHub Advanced Security), Microsoft Entra, Microsoft Intune, Microsoft Purview, Agent 365 SDK, Windows 365 for Agents, Microsoft Execution Container (MXC) SDK, GitHub Copilot, OpenAI Codex, Claude Code, OpenClaw
Published	2026-06-02T17:15:18+00:00
Discovery Source	Rss:T1 Threatintel

Executive Summary

Microsoft has moved its Multi-model Agentic Scanning Harness (MDASH) into expanded enterprise preview, positioning AI agents as the primary triage layer for code security findings across the software development lifecycle. The platform targets a real and growing problem: AI-assisted development pipelines generate code at a pace that outstrips human security review capacity, creating blind spots in access control, input validation, and permission assignment. For CISOs, MDASH represents both a tool to evaluate and a governance question to answer, specifically, how much triage authority an AI agent should hold in a production security pipeline.

Technical Analysis

MDASH enters enterprise preview as an integrated platform spanning Microsoft Defender, GitHub Code Security (GitHub Advanced Security), Microsoft Purview, and the Agent 365 SDK. Its architectural premise is direct: replace static, unvalidated vulnerability lists with exploitability-validated findings generated by AI agents operating inside sandboxed execution environments. The Microsoft Execution Container (MXC) SDK provides that sandbox layer; Windows 365 for Agents supplies a managed execution surface for agent workloads.

The CWE patterns MDASH targets, CWE-20 (improper input validation), CWE-284 (improper access control), CWE-200 (information exposure), CWE-94 (code injection), and CWE-732 (incorrect permission assignment), align directly with the vulnerability classes most commonly introduced by AI code generation tools. Research

from multiple AppSec organizations has documented that large language models producing code frequently mishandle input boundaries and default to overly permissive access patterns, making these five CWE classes a credible priority for any shop running GitHub Copilot, OpenAI Codex, Claude Code, or similar tooling.

The MITRE ATT&CK techniques associated with the platform's threat model span T1059 (command and scripting interpreter), T1078 (valid accounts), T1190 (exploit public-facing application), T1526 (cloud service discovery), T1530 (data from cloud storage), T1552 (unsecured credentials), and T1650 (acquire access). This mapping indicates MDASH is designed to detect not just code-quality defects but patterns that map to initial access and credential exposure in AI-heavy pipelines, a more operationally relevant framing than traditional SAST output.

The strategic tension security teams must evaluate is the agent-in-the-loop model itself. MDASH does not just surface findings; it performs triage decisions. That means the platform's own accuracy, false-negative rate, and susceptibility to prompt injection or adversarial input becomes a security concern independent of the code it reviews. Security teams adopting MDASH inherit a new attack surface: the agents doing the scanning. Microsoft's MXC sandbox addresses some of this risk at the execution layer, but governance questions around agent permissions (AC-6, least privilege), agent audit trails (AU-2, AU-12), and agent identity (IA controls) remain for each organization to resolve within their own policy frameworks.

The platform's timing reflects a broader industry shift. DevSecOps programs that staffed for human-reviewed static analysis pipelines are now watching AI-generated code volumes outpace review capacity. MDASH is Microsoft's answer to that throughput problem. Whether security teams accept that answer depends on how much confidence they can place in the agent's triage logic, and what controls they implement to validate it.

Action Checklist

1. Step 1: Assess exposure. Inventory whether your organization uses MDASH components or AI coding tools: Microsoft Defender, GitHub Advanced Security, Microsoft Purview, Agent 365 SDK, Windows 365 for Agents, MXC SDK, GitHub Copilot, OpenAI Codex, or Claude Code. Verify whether your development pipelines generate AI-assisted code at scale (NIST CM-8, CIS 1.1, CIS 2.1). Assess integration points for agent identity provisioning through Microsoft Entra if applicable.
2. Step 2: Review controls for AI agent governance. Before onboarding MDASH or any agentic security tooling, verify that least-privilege policies extend to agent identities (NIST AC-6), agent actions are logged with sufficient fidelity for review (NIST AU-2, AU-12, CIS 8.2), and agent execution environments are isolated from production systems. Confirm MFA is enforced on all accounts with pipeline access (CIS 6.3, CIS 6.5).
3. Step 3: Update threat model. Incorporate the five CWE classes MDASH targets (CWE-20, CWE-284, CWE-200, CWE-94, CWE-732) into your secure development threat register, specifically tagging findings that originate from AI-generated code. Map associated ATT&CK techniques T1059, T1078, T1190, T1526, T1530, T1552, and T1650 to your detection coverage matrix.
4. Step 4: Define agent authority boundaries. Establish organizational policy governing how much autonomous triage authority AI agents may hold in your security pipeline before human review is required (NIST AC-5, separation of duties). Document exceptions and require sign-off at the security architecture level before MDASH or equivalent tooling is granted production access.
5. Step 5: Monitor developments. Track Microsoft's MDASH enterprise preview release notes, MXC SDK security advisories, and any third-party research on agentic scanner bypass techniques. Register for Microsoft Security Blog updates and GitHub Security advisories relevant to GitHub Advanced Security

integration.

IR / Forensic Enrichment

Triage Priority	STANDARD
Escalation Criteria	Escalate to urgent if your organization is actively onboarding MDASH into a production pipeline that processes code with access to PII, PHI, or financial data, OR if third-party research documents a reproducible technique for injecting adversarial instructions into code comments to cause MDASH to suppress CWE-94 or CWE-284 findings — either condition materially increases the blast radius from medium to high given T1059 and T1190 coverage gaps.
Recovery Notes	If MDASH is already deployed and you identify that the agent suppressed findings that should have been reviewed, immediately export the full MDASH triage decision log for the affected timeframe, rerun GitHub Advanced Security default queries against all repositories the agent touched, and treat any CWE-94 or CWE-284 findings in AI-generated code as unreviewed regardless of prior agent disposition. Monitor GitHub Advanced Security alert volume and auto-close rates for a minimum of 30 days post-remediation to detect any drift back toward unsupervised agent triage. Verify that all Entra service principals associated with MDASH or Agent 365 SDK have been scoped to least-privilege API permissions before restoring agent execution authority.
Forensic Artifacts	Microsoft Entra Audit Logs (category: ApplicationManagement) — captures service principal creation, permission grant events, and OAuth token issuances for MDASH and Agent 365 SDK identities; adversarial pivot via T1078 (Valid Accounts) against a compromised agent identity would appear here as anomalous token grants from unexpected IP ranges. | GitHub Advanced Security Code Scanning Alert History (API: GET /repos/{owner}/{repo}/code-scanning/alerts?state=dismissed) — the dismissed alerts endpoint specifically reveals findings the MDASH agent auto-closed; a bypass technique would manifest as a spike in auto-dismissed CWE-94 or CWE-284 alerts coinciding with introduction of adversarially crafted code comments. | GitHub Copilot Usage Telemetry (API: GET /orgs/{org}/copilot/usage) — links specific file commits to Copilot suggestion acceptance events, enabling you to isolate which code blocks are AI-generated versus human-written; critical for scoping which files require mandatory human review regardless of MDASH disposition. | MXC SDK Execution Container Logs — if Windows 365 for Agents or MXC SDK is deployed, container stdout/stderr logs capture agent task execution traces; T1059 abuse within a container would appear as unexpected shell invocations or process spawning within the execution environment logs, reviewable via Azure Monitor Container Insights if Log Analytics is configured. | Microsoft Purview Audit Log (Unified Audit Log, workload: 'SecurityComplianceCenter') — records any MDASH-driven policy changes, data classification overrides, or compliance alert suppressions that the agent executed autonomously; T1530 (Data from Cloud Storage) lateral movement following an agent compromise would leave data access events traceable here.

Per-Action IR Details

Step 1: Assess exposure — inventory whether your organization runs any of the affected Microsoft products or AI coding tools: Microsoft Defender, GitHub Advanced Security, Microsoft Entra, Microsoft Intune, Microsoft Purview, Agent 365 SDK, Windows 365 for Agents, MXC SDK, GitHub Copilot, OpenAI Codex, or Claude Code. Verify whether your development pipelines generate AI-assisted code at scale (NIST CM-8, CIS 1.1, CIS 2.1).

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: Establishing IR capability requires knowing which systems and tools are in scope before an incident occurs; inventory gaps in AI-assisted pipeline tooling are a pre-incident readiness failure.

Controls: NIST CM-8 (System Component Inventory), NIST CM-7 (Least Functionality), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory), CIS 2.1 (Establish and Maintain a Software Inventory)

Compensating: Run 'az ad app list --output table' (Azure CLI) to enumerate Entra-registered applications, including any Agent 365 SDK or MXC SDK service principals. Use 'gh api /orgs/{org}/installations' to list GitHub Apps with Advanced Security or Copilot permissions across your GitHub organization. Export results to CSV and diff against your last known-good software inventory. A two-person team can complete this in under two hours using only free Microsoft and GitHub CLI tooling.

Evidence: Before inventorying, capture a point-in-time snapshot: export Microsoft Entra audit logs (Azure Portal > Entra ID > Audit Logs, category 'ApplicationManagement') to preserve any recent service principal creation events tied to MDASH or Agent 365 SDK onboarding. Pull GitHub Advanced Security code scanning alert history via 'gh api /repos/{owner}/{repo}/code-scanning/alerts' to establish a pre-MDASH baseline of AI-flagged findings. These snapshots document the pre-assessment state and are inadmissible if collected after changes are made.

Step 2: Review controls for AI agent governance — before onboarding MDASH or any agentic security tooling, verify that least-privilege policies extend to agent identities (NIST AC-6), agent actions are logged with sufficient fidelity for review (NIST AU-2, AU-12, CIS 8.2), and agent execution environments are isolated from production systems. Confirm MFA is enforced on all accounts with pipeline access (CIS 6.3, CIS 6.5).

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: Preventing incidents requires that agent identities in MDASH and MXC SDK execution containers are governed with the same rigor as human privileged accounts; absent controls here, a compromised agent becomes an unmonitored privileged actor in the pipeline.

Controls: NIST AC-6 (Least Privilege), NIST AC-5 (Separation of Duties), NIST AU-2 (Event Logging), NIST AU-12 (Audit Record Generation), NIST IA-2 (Identification and Authentication — Organizational Users), CIS 6.3 (Require MFA for Externally-Exposed Applications), CIS 6.5 (Require MFA for Administrative Access), CIS 8.2 (Collect Audit Logs)

Compensating: In Microsoft Entra, navigate to App Registrations > [MDASH or Agent 365 service principal] > API Permissions and audit for overly broad Microsoft Graph scopes (e.g., 'Directory.ReadWrite.All' is a red flag; 'Code.Read' is acceptable). Use the free Entra 'Conditional Access What If' tool to confirm MFA policy applies to service accounts with pipeline access. For audit log coverage without a SIEM, configure Entra Diagnostic Settings to stream to a Log Analytics Workspace (pay-per-GB, low cost for small teams) and enable GitHub Advanced Security audit log streaming to the same workspace using the GitHub native integration.

Evidence: Pull Microsoft Entra sign-in logs filtered on service principal authentication events for MDASH-related app registrations (Azure Portal > Entra ID > Sign-in Logs > Service Principal Sign-ins). Look specifically for token issuances from non-MFA-enforced conditional access policies and for any refresh token grants with 'offline_access' scope, which would allow persistent agent access without re-authentication. Also review GitHub Advanced Security audit logs (Settings > Security > Audit Log, filter action:'code_scanning.*') for any scanning configuration changes made by automated identities rather than human accounts.

Step 3: Update threat model — incorporate the five CWE classes MDASH targets (CWE-20, CWE-284, CWE-200, CWE-94, CWE-732) into your secure development threat register, specifically tagging findings that originate from AI-generated code. Map associated ATT&CK techniques T1059, T1078, T1190, T1526, T1530, T1552, and T1650 to your detection coverage matrix.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis: Integrating MDASH-targeted CWE classes and associated ATT&CK techniques into the detection coverage matrix ensures that AI-generated code vulnerabilities (particularly CWE-94 code injection and CWE-284 improper access control) are surfaced by existing detection tooling before they reach production.

Controls: NIST RA-3 (Risk Assessment), NIST SI-10 (Information Input Validation), NIST SA-11 (Developer Testing and Evaluation), NIST CA-7 (Continuous Monitoring), CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Import the MITRE ATT&CK Navigator (free, browser-based at attack.mitre.org/resources/attack-navigator) and create a layer highlighting T1059 (Command and Scripting Interpreter), T1078 (Valid Accounts), T1190 (Exploit Public-Facing Application), T1526 (Cloud Service Discovery), T1530 (Data from Cloud Storage), T1552 (Unsecured Credentials), and T1650 (Acquire Access). For each technique, note whether your current tooling (even Sysmon + Windows Event Log) provides any detection signal. For CWE-94 (Code Injection) in AI-generated code specifically, deploy a YARA rule scanning your repository clones for patterns like 'eval(', 'exec(', 'subprocess.call(shell=True' in Python files committed by Copilot-attributed commit authors.

Evidence: Before updating the threat model, extract the current GitHub Advanced Security code scanning alert backlog filtered to AI-generated files — if your team uses Copilot, identify commits where the GitHub metadata shows Copilot suggestion acceptance (available in Copilot usage telemetry via 'GET /orgs/{org}/copilot/usage' API endpoint). Cross-reference open GHAS alerts against CWE-20, CWE-284, CWE-200, CWE-94, and CWE-732 to establish a pre-MDASH baseline of existing exposure in AI-assisted code. This baseline is the evidentiary anchor for demonstrating whether MDASH improves or misses findings relative to your current posture.

Step 4: Define agent authority boundaries — establish organizational policy governing how much autonomous triage authority AI agents may hold in your security pipeline before human review is required (NIST AC-5, separation of duties). Document exceptions and require sign-off at the security architecture level before MDASH or equivalent tooling is granted production access.

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: Defining human-in-the-loop thresholds for MDASH agent triage authority is an IR preparedness requirement; if an AI agent autonomously dismisses a true-positive security finding (e.g., CWE-94 in production-bound code), the incident response team inherits a delayed detection problem with no audit trail of the dismissal decision.

Controls: NIST AC-5 (Separation of Duties), NIST AC-6 (Least Privilege), NIST IR-2 (Incident Response Training), NIST SA-10 (Developer Configuration Management), NIST PL-2 (System Security and Privacy Plans)

Compensating: Draft a one-page 'AI Agent Authority Matrix' in a shared document defining three tiers: (1) Auto-close permitted — informational findings with CVSS < 3.0 and no CWE-94/CWE-284 tag; (2) Human review required — medium findings or any AI-generated code flagged for CWE-20, CWE-732, or T1078-adjacent patterns; (3) Security architect sign-off required — any finding touching authentication logic, secrets handling, or cloud service permission assignment. Store this as a versioned policy file in your GitHub repository under '.github/SECURITY_AGENT_POLICY.md' so changes are tracked in git history. No commercial tooling required.

Evidence: Before finalizing authority boundaries, pull the MDASH or GitHub Advanced Security triage decision log if your organization is already in the enterprise preview — request the audit export from your Microsoft account team, as this is not yet surfaced in standard portal views. Specifically look for any instances where MDASH auto-dismissed a finding subsequently reopened by a human reviewer, which would indicate the agent's false-negative rate on your specific codebase. This data directly informs where to draw the human-review threshold in the authority matrix.

Step 5: Monitor developments — track Microsoft's MDASH enterprise preview release notes, MXC SDK security advisories, and any third-party research on agentic scanner bypass techniques. Register for Microsoft Security Blog updates and GitHub Security advisories relevant to GitHub Advanced Security integration.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: Continuous monitoring of MDASH preview release notes and emerging agentic bypass research is analogous to the lessons-learned and intelligence-sharing function; new attack techniques against AI triage layers (e.g., adversarial prompt injection into code comments to manipulate MDASH findings) represent evolving threat intelligence that must feed back into detection and policy updates.

Controls: NIST SI-5 (Security Alerts, Advisories, and Directives), NIST RA-3 (Risk Assessment), NIST CA-7 (Continuous Monitoring), NIST IR-4 (Incident Handling), CIS 7.1 (Establish and Maintain a Vulnerability Management

Process)

Compensating: Configure a free GitHub Action in your security repository using the 'actions/github-script' action to query the GitHub Security Advisories API weekly for any advisories affecting 'github-advanced-security' or 'copilot' ecosystem packages. Set up an RSS feed monitor (free tools: Feedly, inoreader) on the Microsoft Security Response Center (MSRC) blog filtered for 'MDASH', 'MXC SDK', and 'Agent 365' terms. Subscribe to the CISA Known Exploited Vulnerabilities (KEV) catalog RSS feed — if any MDASH-adjacent CVE is added, KEV addition is your escalation trigger for immediate re-evaluation of agent authority boundaries.

Evidence: Maintain a running log of MDASH release notes deltas — specifically document any changes to the scanner's finding suppression logic, confidence thresholds, or supported CWE classes between preview versions. If a future MDASH update changes how CWE-94 or CWE-284 findings are scored, your baseline audit log (captured in Steps 1-2) becomes the comparison point for determining whether the update introduced regression in detection coverage. Archive each release note version in your git repository as a dated markdown file under '.github/mdash-release-history/'.

Detection Guidance

Security teams evaluating or piloting MDASH should instrument the following detection and audit points:

Agent identity and privilege monitoring: Enable logging for all agent account activity under Microsoft Entra (formerly Azure AD). Alert on agent identities acquiring permissions beyond their provisioned scope, this maps to CWE-732 and ATT&CK T1078 (valid accounts). Apply NIST AC-6 (least privilege) controls to agent service principals and review NIST AU-12 audit record generation for agent-initiated actions.

Pipeline injection surface: AI-generated code touching input handling logic should be flagged for CWE-20 review before merge. Detection rule: any function introduced by an AI coding assistant that processes external input without explicit validation should trigger a manual review gate. This addresses ATT&CK T1059 (command and scripting interpreter) and T1190 (exploit public-facing application).

Cloud storage and credential exposure: In pipelines using Agent 365 or MXC SDK, monitor for agent processes accessing cloud storage outside their designated scope (ATT&CK T1530) or reading secrets from environment variables and configuration files (T1552, maps to CWE-200). Enforce credential rotation policies (NIST IA-4, IA-5) and user account access controls (NIST AC-2, AC-3) at the pipeline credential layer.

Audit log integrity for agent actions: Verify that MDASH agent actions generate tamper-evident audit records (NIST AU-9). Any gap in agent-action logging during a scanning session should trigger an alert; legitimate agentic scanners should not require log suppression to function.

Sandbox escape indicators: For organizations using the MXC SDK, monitor for agent processes attempting network connections outside declared sandbox boundaries, file system access outside the container's declared scope, or process spawning that does not match the agent's declared execution manifest. Apply NIST SI-7 (software, firmware, and information integrity) controls to container configuration files.

Third-party AI tool surface: GitHub Copilot, OpenAI Codex, and Claude Code all contribute to the code corpus MDASH reviews. Maintain a software inventory of which AI coding assistants are authorized (CIS 2.1, CIS 2.3) and flag code commits that cannot be attributed to a specific tool or developer.

Framework Mappings

MITRE-ATTACK

- **T1526** — Cloud Service Discovery

- **T1552** — Unsecured Credentials
- **T1530** — Data from Cloud Storage
- **T1078** — Valid Accounts
- **T1059** — Command and Scripting Interpreter
- **T1190** — Exploit Public-Facing Application
- **T1650** — Acquire Access

NIST-800-53R5

- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-7** — Software, Firmware, and Information Integrity
- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **SI-10** — Information Input Validation
- **AC-3** — Access Enforcement
- **SC-28** — Protection of Information at Rest

OWASP-TOP10-2021

- **A03:2021** — Injection
- **A01:2021** — Broken Access Control

CIS-V8

- **16.10** — Apply Secure Design Principles in Application Architectures
- **3.3** — Configure Data Access Control Lists
- **6.1** — Establish an Access Granting Process
- **6.2** — Establish an Access Revoking Process
- **8.2** — Collect Audit Logs

HIPAA-SECURITY

- **164.312(a)(1)** — Access Control

ISO-27001-2022

- **A.8.26** — Application security requirements
- **A.8.8** — Management of technical vulnerabilities

SOC2-TSC

- **CC6.1** — The entity implements logical access security software, infrastructure, and architectures over protected information assets

NIST-CSF-2

- **DE.CM-01** — Networks and network services are monitored

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1526	Cloud Service Discovery	Discovery
T1552	Unsecured Credentials	Credential-Access
T1530	Data from Cloud Storage	Collection
T1078	Valid Accounts	Defense-Evasion
T1059	Command and Scripting Interpreter	Execution
T1190	Exploit Public-Facing Application	Initial-Access
T1650	Acquire Access	Resource-Development

Sources

Source	URL	Tier
Microsoft Security Blog	https://www.microsoft.com/en-us/security/blog/2026/06/02/microsoft-...	T1
	https://www.microsoft.com/en-us/security/blog/2026/06/02/microsoft-...	T1
	https://www.zdnet.com/article/build-2026-mdash-security-ai-agents/	T3
	https://www.helpnetsecurity.com/2026/06/02/cybersecurity-jobs-avail...	T3
Windows platform security for AI agents - Windows Developer Blog	https://blogs.windows.com/windowsdeveloper/2026/06/02/windows-platf...	T3

Generated 2026-06-03 06:52 UTC by TJS Security Command Center