INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-25 06:49 UTC

Edgecution: Native Messaging API Weaponized to Break Browser Sandbox in Ransomware Delivery Chain

THREAT CAMPAIGN | HIGH | CVSS 7.5

SCC Item ID	SCC-CAM-2026-0561
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	7.5
Affected Products	Microsoft Edge (extensions-enabled), Microsoft Teams, Microsoft 365, Outlook, Windows (AutoHotKey, PowerShell, batch scripting), Python 3.13.3
Published	2026-06-24T16:58:22
Discovery Source	Rss

Executive Summary

An initial access broker linked to the Payouts Kings ransomware group has deployed a malicious Microsoft Edge extension, named 'Edgecution,' that abuses a legitimate browser-to-host communication API to escape the browser sandbox and install a persistent backdoor on Windows systems. The attack begins with social engineering via Microsoft Teams, where attackers impersonate IT support to convince employees to install the extension. Organizations using Microsoft Edge with extensions enabled, Microsoft Teams, and Microsoft 365 environments face direct risk of ransomware staging and lateral movement with potential for full network compromise.

Technical Analysis

The Edgecution campaign, attributed by Zscaler ThreatLabz to an unnamed initial access broker (IAB) supporting the Payouts Kings ransomware group, weaponizes the Chrome Native Messaging protocol (a signed, trusted browser-to-host interoperability API) to bridge the Microsoft Edge browser sandbox. No CVE is assigned; the technique exploits by-design API functionality rather than a discrete software vulnerability. Relevant CWEs: CWE-494 (Download of Code Without Integrity Check), CWE-284 (Improper Access Control), CWE-693 (Protection Mechanism Failure). The attack chain: (1) Vishing via Microsoft Teams, attacker impersonates IT support (T1566, T1566.002, T1566.004); (2) victim installs malicious Edge extension (T1176); (3) extension invokes Native Messaging to communicate with a host-side native application, bypassing the browser sandbox boundary; (4) a Python 3.13-based backdoor is dropped (T1105), supporting PowerShell

invocation (T1059.001), arbitrary Python execution (T1059.006), shell command execution, scheduled persistence (T1547, T1053.005), data collection and exfiltration (T1082, T1057, T1560.001, T1041), and C2 over HTTP/S (T1071, T1071.001). AutoHotKey and batch scripting are also used in the chain. No patch exists, the Native Messaging API is functioning as designed. Mitigation requires policy-level controls on extension installation and Native Messaging host registration.

Action Checklist

1. Step 1: Containment. Immediately audit all Microsoft Edge browser extension installations across the enterprise using Microsoft Intune or Group Policy. Block unauthorized extension installation by enforcing an allowlist via Group Policy (Computer Configuration > Administrative Templates > Microsoft Edge > Extensions > Control which extensions cannot be installed) or Intune (Settings Catalog > Microsoft Edge > Extension Management). Disable or restrict Native Messaging host registration on endpoints where Edge extensions are not operationally required. Reference: Zscaler ThreatLabz Edgecution advisory (https://www.zscaler.com/blogs/security-research/payouts-king-ransomware-initial-access-broker-deploys-new-edgecution). NIST AC-3 (Access Enforcement), NIST AC-6 (Least Privilege). CIS 2.3 (Address Unauthorized Software).
2. Step 2: Detection. Query endpoint logs for Native Messaging host registration events (Windows Registry path: HKCU\Software\Google\Chrome\NativeMessagingHosts and HKCU\Software\Microsoft\Edge\NativeMessagingHosts). Hunt for Python 3.13 processes spawned as children of msedge.exe or browser helper processes. Search EDR telemetry for AutoHotKey and batch script execution originating from browser process trees (T1204.002). Review Microsoft Teams logs for inbound contacts from external or unverified accounts initiating IT support conversations (T1566.004). Flag PowerShell invocations (T1059.001) and scheduled task creation (T1053.005) following Edge process activity. NIST AU-2 (Event Logging), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST SI-4 (Information System Monitoring). CIS 8.2 (Collect Audit Logs).
3. Step 3: Eradication. Remove any unauthorized Edge extensions identified in containment. Purge Native Messaging host manifest files registered outside approved software inventory (check both HKCU and HKLM registry hives and %APPDATA%\Microsoft\Edge\User Data paths). Terminate and remove any Python backdoor processes and associated persistence mechanisms: scheduled tasks (T1053.005), Run key entries (T1547), and any dropped Python scripts or AutoHotKey executables. Block Python interpreter execution on endpoints where it is not operationally required using application control policy. NIST CM-2 (Baseline Configuration). CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.3 (Address Unauthorized Software), CIS 4.6 (Securely Manage Enterprise Assets and Software).
4. Step 4: Recovery. Validate that Native Messaging host registrations are clean on all remediated endpoints. Confirm no residual scheduled tasks or autorun entries remain associated with the campaign. Re-image endpoints where backdoor execution is confirmed rather than relying solely on artifact removal. Monitor post-remediation for re-infection indicators: renewed Python child processes under Edge, new Native Messaging host registrations, and outbound HTTP/S C2 traffic patterns to previously unseen destinations (T1071.001, T1041). Verify Microsoft Teams external communication settings are restricted to block unsolicited external contact. NIST AU-6 (Audit Record Review, Analysis, and Reporting).
5. Step 5: Post-Incident. Conduct a control gap review focused on browser extension governance: most organizations lack a formal allowlist for permitted Edge extensions, which this campaign directly exploited. Implement formal Native Messaging host registration auditing as a standing detection rule. Reinforce employee awareness training specifically targeting Teams-based IT support impersonation as a social

engineering vector. Review whether Python runtime is inventoried and access-controlled across the endpoint fleet. Map gaps to NIST AC-6 (Least Privilege), CIS 5.4 (Restrict Administrator Privileges to Dedicated Administrator Accounts), CIS 7.1 (Establish and Maintain a Vulnerability Management Process). Verify MFA is enforced on all Teams and M365 accounts per CIS 6.3 and CIS 6.5.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate immediately to senior IR leadership, legal counsel, and executive stakeholders if forensic analysis confirms Python backdoor execution on any endpoint, any lateral movement from the initially compromised host to systems holding PII, PHI, or payment card data, or any evidence of ransomware staging activity attributable to the Payouts Kings group, as these conditions trigger breach notification obligations under applicable data protection regulations and indicate the campaign has progressed beyond initial access.
Recovery Notes	Re-image all endpoints where Python backdoor execution is confirmed rather than relying on artifact removal, given the Payouts Kings group's sophistication and the risk of undetected persistence mechanisms beyond the identified scheduled tasks and Run key entries. Post-remediation, maintain enhanced monitoring of Edge NativeMessagingHosts registry paths, Python process ancestry under msedge.exe, and outbound connections to newly-seen destinations for a minimum of 30 days, as IAB-delivered campaigns frequently include secondary backdoors activated after the primary payload is removed. Verify Microsoft Teams External Access policy is set to block unsolicited external federation contacts enterprise-wide before returning any affected users to normal operations, as this closes the primary Edgecution delivery channel.
Forensic Artifacts	Edge extension manifest and background script files at '%LOCALAPPDATA%\Microsoft\Edge\User Data\Default\Extensions\[Edgecution ExtensionID]\' — these contain the Native Messaging host configuration embedded by the malicious extension and are the primary evidence of the sandbox escape mechanism. | Windows Registry keys at 'HKCU\Software\Microsoft\Edge\NativeMessagingHosts\[malicious host name]' and the corresponding JSON manifest file it points to outside approved directories — these document the rogue Native Messaging host registration that enabled browser-to-host IPC abuse. | Python 3.13.3 backdoor script files and any compiled Python bytecode (.pyc) dropped under '%APPDATA%', '%TEMP%', or '%USERPROFILE%\Downloads', preserving the C2 configuration, persistence logic, and any ransomware staging functionality embedded in the Payouts Kings delivery chain. | Microsoft-Windows-TaskScheduler/Operational Event Log entries (Event IDs 106, 200, 201) and associated XML task definition files under '%SYSTEMROOT%\System32\Tasks\' showing scheduled tasks created with Python interpreter or AutoHotKey executable paths, establishing persistence timeline relative to the Edge process activity. | Microsoft Teams Unified Audit Log entries (exported from M365 Compliance Center) documenting the inbound external federation contact used to deliver the Edgecution social engineering lure, including sender identity, timestamp, and message content, which establish the initial access vector and support attribution to the Payouts Kings IAB.

Per-Action IR Details

Step 1: Containment — Immediately audit all Microsoft Edge browser extension installations across the enterprise using Microsoft Intune or Group Policy. Block unauthorized extension installation by enforcing an allowlist via Edge ExtensionInstallAllowlist and ExtensionInstallBlocklist Group Policy Objects. Disable or

restrict Native Messaging host registration on endpoints where Edge extensions are not operationally required. Reference: Zscaler ThreatLabz Edgecution advisory (https://www.zscaler.com/blogs/security-research/payouts-king-ransomware-initial-access-broker-deploys-new-edgecution). NIST AC-3 (Access Enforcement), NIST AC-6 (Least Privilege). CIS 2.3 (Address Unauthorized Software).

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy: isolate the affected component (Edge extension ecosystem and Native Messaging host registration surface) to prevent the Edgecution backdoor from propagating or establishing additional C2 footholds before eradication begins.

Controls: NIST AC-3 (Access Enforcement), NIST AC-6 (Least Privilege), CIS 2.3 (Address Unauthorized Software)

Compensating: Without Intune, enumerate installed Edge extensions on each endpoint by parsing the JSON manifests under '%LOCALAPPDATA%\Microsoft\Edge\User Data\Default\Extensions\' using a PowerShell loop: 'Get-ChildItem $env:LOCALAPPDATA\Microsoft\Edge\User Data\Default\Extensions -Recurse -Filter manifest.json | Select-String -Pattern name'. Cross-reference extension IDs against your approved list and flag any unknown IDs. Disable Native Messaging host registration manually by deleting or renaming registry keys under 'HKCU\Software\Microsoft\Edge\NativeMessagingHosts' on affected hosts using 'reg delete' commands. A two-person team can script this across a small fleet using PsExec or a simple PowerShell remoting loop.

Evidence: Before modifying any registry keys or uninstalling the Edgecution extension, capture the following volatile state: (1) Full memory image of the msedge.exe process tree using WinPmem or Magnet RAM Capture to preserve any in-memory backdoor payload or injected Python interpreter state. (2) Live registry export of 'HKCU\Software\Microsoft\Edge\NativeMessagingHosts' and 'HKLM\Software\Microsoft\Edge\NativeMessagingHosts' using 'reg export' to preserve the Native Messaging host manifest paths and executable references before deletion. (3) Snapshot of all running processes and their parent-child relationships using 'Get-Process | Select-Object Id,ProcessName,MainWindowTitle | Out-File' and 'Get-WmiObject Win32_Process | Select-Object ProcessId,ParentProcessId,CommandLine | Out-File' to document any Python or AutoHotKey processes currently running under msedge.exe. (4) Copy the full Edge extension directory '%LOCALAPPDATA%\Microsoft\Edge\User Data\Default\Extensions\[ExtensionID]\' to forensic storage before removal, as this contains the malicious extension manifest, background scripts, and any native messaging configuration embedded by Edgecution.

Step 2: Detection — Query endpoint logs for Native Messaging host registration events (Windows Registry path: HKCU\Software\Google\Chrome\NativeMessagingHosts and HKCU\Software\Microsoft\Edge\NativeMessagingHosts). Hunt for Python 3.13.3 processes spawned as children of msedge.exe or browser helper processes. Search EDR telemetry for AutoHotKey and batch script execution originating from browser process trees (T1204.002). Review Microsoft Teams logs for inbound contacts from external or unverified accounts initiating IT support conversations (T1566.004). Flag PowerShell invocations (T1059.001) and scheduled task creation (T1053.005) following Edge process activity. NIST AU-2 (Event Logging), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST SI-4 (no mapped control — SI-4 not present in provided knowledge base reference; flag for analyst review). CIS 8.2 (Collect Audit Logs).

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis: correlate process ancestry, registry modification, and Teams communication telemetry to confirm Edgecution extension installation, Native Messaging API abuse, and successful sandbox escape leading to Python backdoor execution.

Controls: NIST AU-2 (Event Logging), NIST AU-6 (Audit Record Review, Analysis, And Reporting), CIS 8.2 (Collect Audit Logs)

Compensating: Without EDR, deploy Sysmon with a configuration that enables Event ID 1 (Process Create) with ParentImage filtering on msedge.exe, and Event ID 13 (Registry Value Set) targeting 'HKCU\Software\Microsoft\Edge\NativeMessagingHosts'. Use the following PowerShell query against Windows Security Event Log to detect Python spawned from Edge: 'Get-WinEvent -LogName Security | Where-Object {$_.Id -eq 4688} | Where-Object {$_.Message -like "*python*" -and $_.Message -like "*msedge*"}'. For Teams log review, export the unified audit log from Microsoft 365 Compliance Center (no SIEM required) filtering on 'MeetingDetail' and

'MessageSent' operations from external federated users. Use Sigma rule 'proc_creation_win_python_spawned_from_browser.yml' (community Sigma repo) as a detection template for manual log correlation.

Evidence: This is an analysis step that does not alter live system state, so order-of-volatility constraints are advisory rather than blocking; however, if analysis confirms active compromise, capture volatile state immediately before proceeding to containment: (1) Windows Security Event Log Event ID 4688 (Process Creation) filtered for python.exe, python3.exe, autohotkey.exe, or cmd.exe with ParentProcessName matching msedge.exe or a browser helper process. (2) Sysmon Event ID 1 logs showing full command-line arguments of any Python 3.13.3 invocation to identify the backdoor script path and C2 configuration. (3) Microsoft Teams Unified Audit Log entries showing inbound external contact initiation — specifically 'External' federation type combined with keywords like 'IT support', 'helpdesk', or 'ticket' in message content. (4) Windows Registry hive artifacts at 'HKCU\Software\Microsoft\Edge\NativeMessagingHosts' showing a manifest path pointing outside '%PROGRAMFILES%' or '%LOCALAPPDATA%\Microsoft\Edge' (legitimate hosts), which indicates Edgecution's rogue Native Messaging host registration. (5) Scheduled Task XML files under '%SYSTEMROOT%\System32\Tasks\' or '%SYSTEMROOT%\SysWOW64\Tasks\' created within the timeframe of the suspected compromise, referencing Python interpreter paths or AutoHotKey executables.

Step 3: Eradication — Remove any unauthorized Edge extensions identified in containment. Purge Native Messaging host manifest files registered outside approved software inventory (check both HKCU and HKLM registry hives and %APPDATA%\Microsoft\Edge\User Data paths). Terminate and remove any Python backdoor processes and associated persistence mechanisms: scheduled tasks (T1053.005), Run key entries (T1547), and any dropped Python scripts or AutoHotKey executables. Block Python interpreter execution on endpoints where it is not operationally required using application control policy. NIST CM (no mapped control from provided CM family entries — flag for analyst review). CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.3 (Address Unauthorized Software), CIS 4.6 (Securely Manage Enterprise Assets and Software).

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication: remove all components of the Edgecution campaign from affected endpoints — the malicious extension, rogue Native Messaging host manifest, Python backdoor binary and scripts, AutoHotKey payloads, and all persistence mechanisms — and verify the environment is clean before recovery begins.

Controls: CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.3 (Address Unauthorized Software), CIS 4.6 (Securely Manage Enterprise Assets and Software)

Compensating: Without application control tooling, block Python execution on non-developer endpoints by renaming or removing the Python interpreter binary ('python.exe', 'python3.exe', 'pythonw.exe') from '%LOCALAPPDATA%\Programs\Python\Python313\' and removing its entry from the system PATH via Group Policy Preferences. Remove rogue scheduled tasks using 'schtasks /delete /tn [TaskName] /f' and verify Run key persistence by querying 'HKCU\Software\Microsoft\Windows\CurrentVersion\Run' and 'HKLM\Software\Microsoft\Windows\CurrentVersion\Run' with 'reg query'. Use ClamAV with a YARA rule targeting the Python backdoor script's known patterns (if IOCs are available from the Zscaler advisory) to scan '%APPDATA%', '%TEMP%', and '%USERPROFILE%\Downloads' for residual payloads.

Evidence: Before terminating Python backdoor processes, killing AutoHotKey executables, or deleting any persistence artifacts, capture the following volatile and forensic state: (1) Full memory acquisition of any running python.exe or pythonw.exe process using WinPmem or ProcDump ('procdump -ma [PID]') to preserve the in-memory backdoor script, decrypted C2 configuration, and any staged ransomware payload components that may not be present on disk. (2) Live network connection table ('Get-NetTCPConnection | Where-Object {$_.State -eq "Established"}' or 'netstat -ano') to capture active C2 IP addresses and ports maintained by the Python backdoor before the process is terminated. (3) Full export of all four persistence registry hives before deletion: 'HKCU\Software\Microsoft\Windows\CurrentVersion\Run', 'HKLM\Software\Microsoft\Windows\CurrentVersion\Run', 'HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce', and 'HKLM\Software\Microsoft\Windows\CurrentVersion\RunOnce'. (4) Forensic copy of any '.py', '.ahk', or '.bat' files found under '%APPDATA%', '%TEMP%', '%USERPROFILE%\Downloads', and '%LOCALAPPDATA%\Temp' before deletion, as these are the Edgecution campaign's primary payload staging directories. (5) Export of the Scheduled

Task XML for any tasks created by the backdoor, preserving trigger logic, execution path, and the identity under which the task runs.

Step 4: Recovery — Validate that Native Messaging host registrations are clean on all remediated endpoints. Confirm no residual scheduled tasks or autorun entries remain associated with the campaign. Re-image endpoints where backdoor execution is confirmed rather than relying solely on artifact removal. Monitor post-remediation for re-infection indicators: renewed Python child processes under Edge, new Native Messaging host registrations, and outbound HTTP/S C2 traffic patterns to previously unseen destinations (T1071.001, T1041). Verify Microsoft Teams external communication settings are restricted to block unsolicited external contact. NIST AU-6 (Audit Record Review, Analysis, and Reporting), D3-LAM (Local Account Monitoring), D3-SFA (System File Analysis).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery: restore affected endpoints to a known-good state, validate completeness of eradication, and establish enhanced monitoring to detect Payouts Kings re-entry or re-infection via the same Teams-based social engineering vector before declaring the incident closed.

Controls: NIST AU-6 (Audit Record Review, Analysis, And Reporting), CIS 4.6 (Securely Manage Enterprise Assets and Software)

Compensating: Without enterprise EDR for post-recovery monitoring, schedule a daily PowerShell script via Task Scheduler on a clean admin host that queries all endpoints for new entries under 'HKCU\Software\Microsoft\Edge\NativeMessagingHosts' and new Python processes with msedge.exe as parent (using 'Get-WmiObject Win32_Process'). Use Wireshark or Windows Firewall logging on remediated endpoints to capture and review outbound connections on ports 443 and 80 to newly-seen destinations for 30 days post-remediation. For Teams external communication restriction, verify the setting in the Microsoft Teams Admin Center under 'External Access' — this requires no additional tooling and blocks the primary Edgecution social engineering delivery channel.

Evidence: Before re-imaging any confirmed-compromised endpoint, ensure the following evidence is preserved and transferred to forensic storage: (1) Full disk image using a write-blocker and forensic imaging tool (e.g., FTK Imager) to preserve the complete artifact chain for potential legal proceedings given the Payouts Kings ransomware group attribution. (2) Final snapshot of all active network connections and DNS cache ('ipconfig /displaydns') to capture any C2 infrastructure contacted during the compromise window. (3) Export of the Windows Event Log channels most relevant to this campaign — Microsoft-Windows-Sysmon/Operational, Security, Microsoft-Windows-TaskScheduler/Operational, and Microsoft-Windows-PowerShell/Operational — before the image is wiped. Recovery actions (re-imaging, restoring from backup, reconfiguring Teams external access) do not themselves require pre-action volatile capture, but all prior volatile collection from eradication must be confirmed complete before the endpoint is returned to production.

Step 5: Post-Incident — Conduct a control gap review focused on browser extension governance: most organizations lack a formal allowlist for permitted Edge extensions, which this campaign directly exploited. Implement formal Native Messaging host registration auditing as a standing detection rule. Reinforce employee awareness training specifically targeting Teams-based IT support impersonation as a social engineering vector. Review whether Python runtime is inventoried and access-controlled across the endpoint fleet. Map gaps to NIST AC-6 (Least Privilege), CIS 5.4 (Restrict Administrator Privileges to Dedicated Administrator Accounts), CIS 7.1 (Establish and Maintain a Vulnerability Management Process). Countermeasures: D3-UAP (User Account Permissions), D3-CH (Credential Hardening), D3-MFA (Multi-factor Authentication) — verify MFA is enforced on all Teams and M365 accounts per CIS 6.3 and CIS 6.5.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: conduct a lessons-learned review to formalize browser extension governance, Native Messaging host auditing, and Teams external-access controls as standing program improvements, and share Edgecution campaign IOCs and TTPs with sector peers to reduce dwell time in future campaigns by this IAB.

Controls: NIST AC-6 (Least Privilege), CIS 5.4 (Restrict Administrator Privileges to Dedicated Administrator Accounts), CIS 6.3 (Require MFA for Externally-Exposed Applications), CIS 6.5 (Require MFA for Administrative Access), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Without a formal GRC platform, document the control gap review in a structured spreadsheet mapping each identified gap (extension allowlist absent, Native Messaging hosts unmonitored, Python uncontrolled, Teams external access open) to the responsible team and a remediation date. Implement the Native Messaging host standing detection rule as a Sigma rule targeting Sysmon Event ID 13 (Registry Value Set) on the 'HKCU\Software\Microsoft\Edge\NativeMessagingHosts' path, shared via the community Sigma repository for peer validation. Deliver the Teams IT-impersonation awareness module as a 10-minute scenario-based exercise using real Edgecution campaign lure examples (sanitized from the Zscaler advisory) — no LMS required.

Evidence: Post-incident activity does not alter live system state, so order-of-volatility constraints do not apply; however, the lessons-learned review should draw on the complete forensic evidence package collected during earlier phases: (1) Timeline reconstruction from Sysmon and Windows Security Event Log showing the precise sequence from Teams message receipt to extension installation to Native Messaging API invocation to Python backdoor execution, to establish accurate dwell time and blast radius metrics. (2) All preserved registry exports, scheduled task XMLs, and Python script artifacts to inform the IOC set shared with sector ISACs and submitted to Microsoft for Edge extension store review. (3) Microsoft Teams Unified Audit Log entries documenting the social engineering conversation chain, which serve as the primary evidence for any HR or legal follow-up regarding the impersonated IT support persona.

Detection Guidance

Primary detection targets: (1) Native Messaging host registration, monitor Windows Registry for new keys under HKCU\Software\Microsoft\Edge\NativeMessagingHosts and HKLM\Software\Microsoft\Edge\NativeMessagingHosts; alert on any registration not in the approved software inventory. (2) Anomalous child process lineage, alert on Python.exe, AutoHotKey.exe, cmd.exe, or powershell.exe spawned as children of msedge.exe or browser extension host processes. (3) Scheduled task creation following browser activity, correlate Windows Event ID 4698 (scheduled task created) with preceding Edge process activity within a short time window. (4) Microsoft Teams external contact abuse, review unified audit logs for inbound Teams messages from external federated users claiming IT support identity, particularly those directing recipients to install software or extensions (T1566.004). (5) Outbound C2, monitor for HTTP/S connections to low-reputation or newly registered domains initiated by Python.exe or cmd.exe processes (T1071.001, T1041). IOC patterns: Python 3.13 binary presence on endpoints where Python is not in the approved software inventory; Native Messaging manifest JSON files referencing non-standard host application paths; batch or AutoHotKey scripts in user-writable directories executed outside business hours. MITRE techniques covered: T1176, T1105, T1059.001, T1059.006, T1053.005, T1547, T1082, T1057, T1560.001, T1041, T1071.001. See Zscaler ThreatLabz Edgecution analysis for additional IOCs and technical details.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`https://www.zscaler.com/blogs/security-research/payouts-king-ransomware-initial-access-broker-deploys-new-edgecution`	Primary source — Zscaler ThreatLabz Edgecution campaign analysis; contains authoritative IOC set (search-retrieved URL, recommend human validation)	MEDIUM

Framework Mappings

MITRE-ATTACK

- **T1105** — Ingress Tool Transfer
- **T1059.001** — PowerShell
- **T1071.001** — Web Protocols
- **T1547** — Boot or Logon Autostart Execution
- **T1566.002** — Spearphishing Link
- **T1053.005** — Scheduled Task
- **T1176** — Software Extensions
- **T1041** — Exfiltration Over C2 Channel
- **T1082** — System Information Discovery
- **T1566** — Phishing
- **T1566.004** — Spearphishing Voice
- **T1190** — Exploit Public-Facing Application
- **T1071** — Application Layer Protocol
- **T1560.001** — Archive via Utility
- **T1057** — Process Discovery
- **T1059.006** — Python
- **T1204.002** — Malicious File

NIST-800-53R5

- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **CM-7** — Least Functionality
- **SI-7** — Software, Firmware, and Information Integrity
- **AT-2** — Literacy Training and Awareness
- **SI-8** — Spam Protection
- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SI-2** — Flaw Remediation
- **CM-3** — Configuration Change Control
- **AC-3** — Access Enforcement
- **CP-9** — System Backup
- **IR-4** — Incident Handling

OWASP-TOP10-2021

- **A08:2021** — Software and Data Integrity Failures
- **A01:2021** — Broken Access Control

CIS-V8

- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **6.1** — Establish an Access Granting Process
- **6.2** — Establish an Access Revoking Process
- **14.2** — Train Workforce Members to Recognize Social Engineering Attacks
- **8.2** — Collect Audit Logs

SOC2-TSC

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

HIPAA-SECURITY

- **164.312(a)(1)** — Access Control
- **164.308(a)(7)(ii)(A)** — Data Backup Plan

NIST-CSF-2

- **RS.MI-01** — Incidents are contained
- **DE.CM-01** — Networks and network services are monitored

ISO-27001-2022

- **A.5.29** — Information security during disruption
- **A.8.8** — Management of technical vulnerabilities

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1105	Ingress Tool Transfer	Command-And-Control
T1059.001	PowerShell	Execution
T1071.001	Web Protocols	Command-And-Control
T1547	Boot or Logon Autostart Execution	Persistence
T1566.002	Spearphishing Link	Initial-Access
T1053.005	Scheduled Task	Execution
T1176	Software Extensions	Persistence
T1041	Exfiltration Over C2 Channel	Exfiltration

Technique ID	Technique Name	Tactic
T1082	System Information Discovery	Discovery
T1566	Phishing	Initial-Access
T1566.004	Spearphishing Voice	Initial-Access
T1190	Exploit Public-Facing Application	Initial-Access
T1071	Application Layer Protocol	Command-And-Control
T1560.001	Archive via Utility	Collection
T1057	Process Discovery	Discovery
T1059.006	Python	Execution
T1204.002	Malicious File	Execution

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/malicious-edge-exten...	T3
Disrupting active exploitation of on-premises SharePoint ... - Microsoft	https://www.microsoft.com/en-us/security/blog/2025/07/22/disrupting...	T1
Unlocking Microsoft 365 Security: How I Automated AI-Powered Risk ...	https://www.youtube.com/watch?v=gyPXlI6GHCo	T3
Edgecution: Malicious Edge Extension Backdoor | ThreatLabz	https://www.zscaler.com/blogs/security-research/payouts-king-ransom...	T3
Microsoft Security Response Center Blog	https://www.microsoft.com/en-us/msrc/blog	T1

Generated 2026-06-25 06:49 UTC by TJS Security Command Center