INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-19 19:00 UTC

ClickOnce Weaponized: Microsoft's Low-Privilege Deployment Framework Becomes a Malware Delivery Channel

SECURITY ANALYSIS | **MEDIUM** | CVSS 5.0

SCC Item ID	SCC-STY-2026-0225
Type	Security Analysis
Severity	MEDIUM
CVSS Base Score	5.0
Affected Products	Microsoft Windows, .NET ClickOnce deployment framework, Visual Studio (all versions supporting ClickOnce)
Discovery Source	Rss:T1 Threatintel

Executive Summary

CrowdStrike researchers have documented how threat actors are weaponizing Microsoft's ClickOnce deployment framework, a legitimate Windows and .NET feature designed for low-friction application installs, to deliver malware without requiring administrative privileges or triggering standard email attachment filters. Because ClickOnce uses trusted Windows infrastructure and operates through normal application update channels, it bypasses many organizations' existing perimeter and endpoint controls with minimal user interaction. This campaign signals a broader attacker shift toward abusing by-design platform functionality, where no patch exists and detection requires deliberate tuning rather than vendor remediation.

Technical Analysis

CrowdStrike's two-part research series (published June 2026) details how ClickOnce, Microsoft's managed deployment framework for .NET applications, has been co-opted as a malware delivery mechanism across Windows environments. The technique is notable precisely because it exploits intended functionality rather than a software defect; no CVE has been assigned, and Microsoft is unlikely to issue a corrective patch.

The attack chain follows a phishing-initiated pattern (MITRE T1566, T1566.002). A target receives either a malicious link or an email carrying a .application or .manifest file, the native file types ClickOnce uses for deployment descriptors (T1204, T1204.002). Most enterprise mail gateways filter executable attachments by extension, but .application files are not universally flagged, creating a reliable perimeter bypass aligned with CWE-693 (Protection Mechanism Failure). A single click triggers the ClickOnce runtime, which fetches and

executes the attacker-staged payload from a remote server under the user's existing privilege context (T1105, T1218).

Because the runtime operates as a legitimate Windows and .NET process, the resulting network traffic resembles normal enterprise application update behavior. Endpoint detection tools not specifically tuned to inspect .application or .manifest file types may not alert on the initial delivery stage. The framework's design also permits persistence mechanisms consistent with T1547, and attackers can use code-signing certificate abuse or masquerading (T1553, T1036) to further reduce the likelihood of detection.

The primary risk driver is not the vulnerability surface of the ClickOnce runtime itself, but the detection gap it exploits. Organizations with mature EDR deployments but no specific rules for ClickOnce-initiated process trees, and mail environments that whitelist non-executable file types, are the most exposed. The attack requires user interaction but imposes only a single-click barrier, a realistic threshold in enterprise phishing scenarios.

CWE-494 (Download of Code Without Integrity Check) is applicable when payload signing is absent or not enforced, though the framework does support optional code-signing validation that most organizations do not mandate through policy. The Microsoft Learn documentation for ClickOnce Security and Deployment confirms that deployment trust decisions can be configured at the enterprise level, a control gap the research explicitly highlights.

Action Checklist

1. Step 1: Assess exposure, audit whether ClickOnce-based application deployment is in active use across your Windows and .NET environment; check Group Policy and AppLocker configurations for any existing ClickOnce controls (NIST CM-7, Least Functionality applies: restrict ClickOnce execution to explicitly authorized applications)
2. Step 2: Review mail gateway rules, confirm that .application, .manifest, and .deploy file extensions are either blocked or subjected to sandboxing and content inspection equivalent to executable attachments at the email perimeter; consult your mail security vendor's documentation to verify inspection coverage
3. Step 3: Tune endpoint detection, work with your EDR vendor to create detection logic for ClickOnce-initiated process trees, specifically parent-child relationships originating from dfsvc.exe or other ClickOnce runtime processes spawning unexpected child processes (NIST SI-4, System Monitoring; CIS 8.2, Collect Audit Logs)
4. Step 4: Audit application allow-listing policy, verify that application control policies (AppLocker, WDAC, or equivalent) restrict unsigned or untrusted ClickOnce payloads; enforce code-signing validation for any ClickOnce deployments your organization authorizes (NIST CM-7, AC-3, Access Enforcement; CIS 2.2, Ensure Authorized Software is Currently Supported)
5. Step 5: Update threat model, add ClickOnce-based phishing delivery (T1566.002, T1204.002, T1218) to your threat register and review phishing simulation scenarios to include non-executable attachment lures
6. Step 6: Communicate findings, brief security leadership and application owners on the ClickOnce exposure; frame it as a detection gap requiring tool tuning, not a patchable vulnerability, so remediation expectations are correctly set
7. Step 7: Monitor developments, CrowdStrike's Part 2 of this research series is the immediate follow-on; track for additional TTPs, published indicators, and any Microsoft policy guidance on enterprise ClickOnce restrictions

IR / Forensic Enrichment

Triage Priority	URGENT
Escalation Criteria	Escalate immediately to IR leadership and legal counsel if Sysmon Event ID 1 or EDR telemetry confirms dfsvc.exe spawning a shell or post-exploitation process on any host, if .application or .manifest attachments are found in delivered mail logs indicating a lure campaign is already active against the organization, or if the ClickOnce application cache at `%LOCALAPPDATA%\Apps\2.0\` contains unsigned or anomalous payloads on more than one endpoint, as this indicates active delivery rather than theoretical exposure.
Recovery Notes	After containment controls are in place (AppLocker or WDAC policy enforced, mail gateway blocking ClickOnce extensions, dfsvc.exe detection rules active), verify that no residual ClickOnce-deployed malware persists by auditing the `%LOCALAPPDATA%\Apps\2.0\` cache on all endpoints and comparing installed application hashes against your authorized software inventory. Monitor Sysmon Event ID 1 for dfsvc.exe process creation for a minimum of 30 days post-containment, as ClickOnce payloads can be triggered by user revisiting a malicious URL or opening a cached .application manifest. Re-run phishing simulation with a ClickOnce lure within 60 days to validate that user awareness and gateway controls are functioning as a layered defense.
Forensic Artifacts	ClickOnce application cache directory: `%LOCALAPPDATA%\Apps\2.0\` — contains all ClickOnce-deployed application binaries, manifests (.application, .manifest files), and DLLs with original filenames and timestamps; malicious payloads delivered via weaponized ClickOnce will reside here and may include obfuscated or renamed executables | Windows Registry key `HKCU:\Software\Microsoft\Windows\CurrentVersion\Uninstall` — ClickOnce applications register under HKCU (not HKLM) without admin rights; each subkey contains DeploymentProviderUrl revealing the C2 or attacker-controlled distribution server hosting the malicious .application manifest | Sysmon Event ID 1 (Process Creation) logs in `Microsoft-Windows-Sysmon/Operational` — specifically entries where ParentImage is `C:\Windows\SysWOW64\dfshim.dll` or `dfsvc.exe` and Image is any shell or interpreter (powershell.exe, cmd.exe, mshta.exe), capturing the exact child process spawned by the weaponized ClickOnce payload | Email gateway logs and quarantine queue filtered for attachments with extensions .application, .manifest, or .deploy — these reveal whether a ClickOnce phishing lure was delivered to users, which recipient accounts were targeted, and the sending domain or IP of the attacker-controlled distribution infrastructure | Windows Security Event ID 4688 (Process Creation with command-line logging enabled) from the Security event log — provides a command-line audit trail of all processes launched under the user context following dfsvc.exe execution, capturing attacker post-exploitation commands (e.g., reconnaissance, lateral movement, payload staging) that occurred after ClickOnce delivered the initial implant without admin privilege escalation

Per-Action IR Details

Step 1: Assess exposure — audit whether ClickOnce-based application deployment is in active use across your Windows and .NET environment; check Group Policy and AppLocker configurations for any existing ClickOnce controls (NIST CM-7 — Least Functionality applies: restrict ClickOnce execution to explicitly authorized applications)

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: establishing the capability to detect and respond before an incident occurs

Controls: NIST AC-3 (Access Enforcement), NIST AC-6 (Least Privilege), CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory), CIS 2.1 (Establish and Maintain a Software Inventory)

Compensating: Without a CMDB or enterprise MDM, run the following on each Windows host: `Get-ChildItem 'HKCU:\Software\Classes\' | Where-Object { $_.Name -like '*clickonce*' }` and `Get-AppLockerPolicy -Effective | Export-AppLockerPolicy -XMLPolicy C:\output\applocker_policy.xml`. Use `reg query 'HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall' /s /f '.application'` to surface ClickOnce-installed apps. Cross-reference results against an authorized software list maintained in a spreadsheet.

Evidence: This step performs read-only enumeration and does not alter live host state. No volatile pre-capture is required before the audit itself. However, document baseline registry state at `HKCU:\Software\Microsoft\Windows\CurrentVersion\Uninstall` and `HKCU:\Apps\2.0\` (the ClickOnce application cache directory) before any subsequent remediation steps, as these paths represent the primary persistence mechanism for ClickOnce-delivered payloads and will be needed for comparison post-incident.

Step 2: Review mail gateway rules — confirm that .application, .manifest, and .deploy file extensions are either blocked or subjected to the same scrutiny as executable attachments at the email perimeter; consult your mail security vendor's documentation to verify inspection coverage

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: configuring preventive controls to reduce incident probability and impact

Controls: NIST AC-4 (Information Flow Enforcement), NIST SI-3 (Malicious Code Protection) — cited from knowledge base as SI family; note SI-3 name is drawn from SP 800-53r5 family structure, CIS 4.4 (Implement and Manage a Firewall on Servers)

Compensating: On Microsoft 365 without a third-party secure email gateway, create a mail flow rule in the Exchange Admin Center: Conditions = 'Attachment extension matches .application OR .manifest OR .deploy'; Action = 'Reject the message with explanation: ClickOnce deployment files are not permitted via email'. For on-premises Exchange, use `New-TransportRule` with `-AttachmentExtensionMatchesWords @('application','manifest','deploy')`. Validate with a test send using a renamed benign file carrying the .application extension.

Evidence: This step modifies gateway configuration and does not touch live host state. Before modifying mail flow rules, export the current rule set (`Get-TransportRule | Export-Clixml rules_baseline.xml`) as a rollback artifact. For forensic purposes, query existing mail logs for historical delivery of .application, .manifest, or .deploy attachments — in Exchange, run `Get-MessageTrackingLog -EventId RECEIVE -Start (Get-Date).AddDays(-90) | Where-Object { $_.Recipients -ne $null }` filtered against attachment metadata to identify whether ClickOnce lures have already entered the environment.

Step 3: Tune endpoint detection — work with your EDR vendor to create detection logic for ClickOnce-initiated process trees, specifically parent-child relationships originating from dfsvc.exe or other ClickOnce runtime processes spawning unexpected child processes (NIST SI-4 — System Monitoring; CIS 8.2 — Collect Audit Logs)

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis: identifying indicators of compromise through monitoring and log analysis

Controls: NIST AU-2 (Event Logging), NIST AU-6 (Audit Record Review, Analysis, and Reporting), NIST AU-12 (Audit Record Generation), CIS 8.2 (Collect Audit Logs)

Compensating: Deploy Sysmon with a configuration that logs Event ID 1 (Process Creation) and Event ID 8 (CreateRemoteThread). Write a Sigma rule targeting `ParentImage: '*\dfsvc.exe'` with `Image` fields matching common post-exploitation binaries (cmd.exe, powershell.exe, mshta.exe, wscript.exe, rundll32.exe). Query with: `Get-WinEvent -LogName 'Microsoft-Windows-Sysmon/Operational' | Where-Object { $_.Message -match 'dfsvc.exe' }`. Additionally, enable Windows Security Event ID 4688 (Process Creation) with command-line auditing via Group Policy (`Computer Configuration > Windows Settings > Security Settings > Advanced Audit Policy > Detailed Tracking`).

Evidence: Before modifying any EDR policy or agent configuration, capture the current process tree on endpoints where dfsvc.exe activity has been observed: run `wmic process where 'name="dfsvc.exe"' get ProcessId,ParentProcessId,CommandLine` and `Get-NetTCPConnection | Where-Object { $_.OwningProcess -eq }` to

record any active network connections attributed to the ClickOnce runtime. Also export `%LOCALAPPDATA%\Apps\2.0\` directory listing and `HKCU:\Software\Microsoft\Windows\CurrentVersion\Uninstall` registry hive before any endpoint agent changes alter timestamps or running process state.

Step 4: Audit application allow-listing policy — verify that application control policies (AppLocker, WDAC, or equivalent) restrict unsigned or untrusted ClickOnce payloads; enforce code-signing validation for any ClickOnce deployments your organization authorizes (NIST CM-7, AC-3 — Access Enforcement; CIS 2.2 — Ensure Authorized Software is Currently Supported)

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy: implementing controls to limit the scope of a threat using application control mechanisms

Controls: NIST AC-3 (Access Enforcement), NIST AC-6 (Least Privilege), CIS 2.2 (Ensure Authorized Software is Currently Supported), CIS 2.3 (Address Unauthorized Software), CIS 4.6 (Securely Manage Enterprise Assets and Software)

Compensating: Without enterprise WDAC management, use AppLocker Publisher rules: open `gpedit.msc > Computer Configuration > Windows Settings > Security Settings > Application Control Policies > AppLocker`. Create a Publisher condition requiring a valid Authenticode signature from a trusted internal CA for any executable originating from `%LOCALAPPDATA%\Apps\2.0\`. For immediate blocking where AppLocker is unavailable, use Software Restriction Policies to set an Additional Rule with Path `%LOCALAPPDATA%\Apps\2.0\*` set to Disallowed. Validate the block by attempting to launch a test .application file and confirming Event ID 8004 (AppLocker block) appears in the Application log.

Evidence: Before pushing any AppLocker or WDAC policy change that would terminate running ClickOnce applications, capture volatile state: run `Get-Process | Where-Object { $_.Path -like '*Apps\2.0*' }` to identify all active ClickOnce-launched processes and record their PIDs, command lines, and network connections via `Get-NetTCPConnection | Where-Object { $_.OwningProcess -in }`. Acquire memory from any suspicious ClickOnce-spawned process using ProcDump (`procdump.exe -ma clickonce_suspect.dmp`) before the policy enforcement kills it. Preserve the `%LOCALAPPDATA%\Apps\2.0\` cache directory contents and file timestamps as forensic evidence of what was deployed.

Step 5: Update threat model — add ClickOnce-based phishing delivery (T1566.002, T1204.002, T1218) to your threat register and review phishing simulation scenarios to include non-executable attachment lures

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: using lessons learned to update detection capabilities and threat models

Controls: CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Maintain a threat register in a shared spreadsheet or wiki. Add three rows for the MITRE ATT&CK techniques referenced (Spearphishing Link, Malicious File, Signed Binary Proxy Execution via ClickOnce), with columns for: detection coverage (Yes/No/Partial), responsible team, and review date. Update phishing simulation platform (GoPhish is free) to include campaigns that deliver a harmless .application file mimicking a legitimate vendor portal to measure user click rates on non-executable lures.

Evidence: This step involves documentation and simulation planning and does not alter live host state, so no volatile pre-capture is required. However, preserve CrowdStrike's published research, any internal incident tickets, and detection alert logs as supporting evidence for the threat model update — these substantiate the risk rating assigned to ClickOnce-based delivery in your register and will be referenced during the next audit or tabletop exercise.

Step 6: Communicate findings — brief security leadership and application owners on the ClickOnce exposure; frame it as a detection gap requiring tool tuning, not a patchable vulnerability, so remediation expectations are correctly set

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: reporting findings to leadership and stakeholders to drive organizational improvement

Controls: NIST AC-1 (Policy and Procedures), NIST AU-1 (Policy and Procedures)

Compensating: Prepare a one-page executive brief using the following structure: (1) What ClickOnce is and why it exists in your environment, (2) How threat actors weaponize it without requiring admin rights or triggering standard AV — reference the CrowdStrike research by name, (3) Current detection gap mapped to specific tooling (e.g., 'our SEG does not inspect .application attachments'), (4) Remediation steps with owners and target dates, (5) Residual risk statement. Deliver via email with read-receipt and retain as evidence of notification for compliance purposes.

Evidence: No live host state is altered by this communication step; no volatile pre-capture is required. Attach to the brief any output collected during Steps 1–4 (AppLocker policy exports, registry enumerations, Sysmon alert counts) as supporting evidence. These artifacts substantiate the exposure claim to leadership and document the baseline state before remediation, which is critical if a ClickOnce-delivered payload is later discovered to have been active during the assessment window.

Step 7: Monitor developments — CrowdStrike's Part 2 of this research series is the immediate follow-on; track for additional TTPs, published indicators, and any Microsoft policy guidance on enterprise ClickOnce restrictions

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: integrating external threat intelligence to continuously improve detection and response posture

Controls: CIS 7.1 (Establish and Maintain a Vulnerability Management Process), CIS 7.2 (Establish and Maintain a Remediation Process)

Compensating: Subscribe to CrowdStrike's blog RSS feed and Microsoft Security Response Center (MSRC) advisory RSS for ClickOnce-relevant updates at no cost. Set a Google Alert for 'ClickOnce malware' and 'dfsvc.exe threat' to capture community reporting. When CrowdStrike Part 2 publishes, immediately extract any new IOCs (file hashes, C2 domains, .application manifest signatures) and convert them to YARA rules or Sysmon keyword filters deployable without a SIEM. Track Microsoft's ClickOnce Group Policy documentation (`gpedit.msc > Computer Configuration > Administrative Templates > Windows Components > Internet Explorer > Security Features > Consistent Mime Handling`) for any new enterprise restriction options.

Evidence: This is an intelligence-monitoring step and does not alter live host state; no volatile pre-capture is required. Maintain a dated change log of IOCs, TTPs, and policy guidance as each update is received — this log serves as the paper trail demonstrating that the organization actively tracked the evolving ClickOnce threat campaign, which is relevant to due-diligence requirements under frameworks such as NIST CSF and supports post-incident review if a breach occurs after this advisory was received.

Detection Guidance

Focus detection engineering on the ClickOnce runtime process (dfsvc.exe) and its behavioral signatures. Key hunting hypotheses and log sources:

1. Process tree anomalies: Hunt for dfsvc.exe spawning cmd.exe, powershell.exe, wscript.exe, mshta.exe, or any network-connected process. This parent-child pattern is consistent with T1218 (System Binary Proxy Execution) and is not expected in legitimate ClickOnce application update workflows. Query endpoint telemetry and EDR process logs.

2. Network fetch from non-enterprise origins: ClickOnce fetches manifests and payloads over HTTP/HTTPS. Correlate dfsvc.exe outbound connections against your approved internal or known-good vendor update servers. Connections to newly registered domains, IP-literal URLs, or infrastructure outside your application vendor baseline warrant investigation (AU-3, Content of Audit Records; AU-6, Audit Record Review, Analysis, and Reporting).

3. Mail gateway file type telemetry: Review logs for inbound .application, .manifest, and .deploy attachments or links resolving to those file types. Flag any delivery from external senders and correlate with downstream endpoint activity.

4. AppLocker and WDAC event logs: Windows event IDs 8003 and 8004 (AppLocker blocked or audited execution) and WDAC equivalent events will surface unauthorized ClickOnce execution attempts if allow-listing is enforced. If these logs show no entries for dfsvc.exe, consider whether allow-listing policy actually covers it.

5. Startup and persistence artifacts: ClickOnce applications install per-user under %LocalAppData%\Apps and can register persistence via the user's run keys or scheduled tasks. Hunt for new entries in HKCU\Software\Microsoft\Windows\CurrentVersion\Run and user-space scheduled tasks created within proximity of a ClickOnce execution event (T1547; NIST SI-4).

6. Unsigned or self-signed manifest inspection: If your environment logs Authenticode validation results, filter for .application or .manifest files with absent, self-signed, or recently issued certificates from untrusted authorities.

D3FEND countermeasures applicable: D3-SFA (System File Analysis) for monitoring ClickOnce install directories; D3-UAP (User Account Permissions) to restrict dfsvc.exe execution scope; D3-LAM (Local Account Monitoring) for persistence activity under user accounts.

Indicators of Compromise

Type	Value	Context	Confidence
TOOL	`dfsvc.exe`	ClickOnce runtime (dfsvc.exe) leveraged via phishing-delivered .application or .manifest files to fetch and execute attacker-staged malware payloads from remote servers under user privilege context	HIGH
URL	`Pending — refer to CrowdStrike blog Part 1 and Part 2 for published indicators`	CrowdStrike's two-part research series may contain C2 URLs, payload staging domains, and file hashes associated with observed ClickOnce abuse campaigns; values were not included in the source material provided	LOW

Framework Mappings

MITRE-ATTACK

- **T1204.002** — Malicious File
- **T1204** — User Execution
- **T1553** — Subvert Trust Controls
- **T1036** — Masquerading
- **T1105** — Ingress Tool Transfer
- **T1566** — Phishing
- **T1218** — System Binary Proxy Execution

- **T1547** — Boot or Logon Autostart Execution
- **T1566.002** — Spearphishing Link

NIST-800-53R5

- **AT-2** — Literacy Training and Awareness
- **CA-7** — Continuous Monitoring
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SC-7** — Boundary Protection
- **SI-8** — Spam Protection
- **CM-7** — Least Functionality
- **SI-7** — Software, Firmware, and Information Integrity
- **CM-3** — Configuration Change Control

OWASP-TOP10-2021

- **A08:2021** — Software and Data Integrity Failures

CIS-V8

- **2.5** — Allowlist Authorized Software
- **2.6** — Allowlist Authorized Libraries
- **14.2** — Train Workforce Members to Recognize Social Engineering Attacks
- **8.2** — Collect Audit Logs

NIST-CSF-2

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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1204.002	Malicious File	Execution
T1204	User Execution	Execution
T1553	Subvert Trust Controls	Defense-Evasion
T1036	Masquerading	Defense-Evasion
T1105	Ingress Tool Transfer	Command-And-Control
T1566	Phishing	Initial-Access
T1218	System Binary Proxy Execution	Defense-Evasion
T1547	Boot or Logon Autostart Execution	Persistence

Technique ID	Technique Name	Tactic
T1566.002	Spearphishing Link	Initial-Access

Sources

Source	URL	Tier
Blog	https://www.crowdstrike.com/en-us/blog/new-abuse-of-the-clickonce-t...	T3
	https://www.crowdstrike.com/en-us/blog/new-abuse-of-the-clickonce-t...	T3
	https://www.crowdstrike.com/en-us/blog/new-abuse-of-the-clickonce-t...	T3
	https://www.crowdstrike.com/en-us/blog/reasons-why-nonprofits-are-t...	T3
ClickOnce Deployment and Security - Visual Studio - Microsoft Learn	https://learn.microsoft.com/en-us/visualstudio/deployment/clickonce...	T1

Generated 2026-06-19 19:00 UTC by TJS Security Command Center