INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-21 18:39 UTC

# ClickOnce Deployment Mechanism Exposed as Malware Delivery Channel: What Defenders Must Know

SECURITY ANALYSIS | MEDIUM | CVSS 5.0

| | |
|---|---|
| SCC Item ID | SCC-STY-2026-0240 |
| Type | Security Analysis |
| Severity | MEDIUM |
| CVSS Base Score | 5.0 |
| Affected Products | Microsoft ClickOnce technology, Windows OS, .NET applications, Visual Studio |
| Discovery Source | Rss:T1 Threatintel |

## Executive Summary

CrowdStrike researchers have documented how Microsoft's ClickOnce deployment technology, a legitimate feature built into Windows and Visual Studio, can be repurposed as a malware delivery channel without requiring administrator privileges or significant user interaction. Because ClickOnce's trust model lacks mandatory integrity verification at the deployment layer, attackers can serve malicious payloads through web servers or network file shares in a way that bypasses conventional installation controls. This research signals a maturing attacker interest in abusing trusted platform mechanisms rather than exploiting discrete software vulnerabilities, a pattern that demands defenders re-examine assumptions about what constitutes a safe application delivery channel.

## Technical Analysis

CrowdStrike's Part 1 analysis of ClickOnce abuse focuses on the technology's structural design rather than a specific CVE, which is a meaningful distinction. ClickOnce was engineered for ease of deployment: applications install without elevated privileges, update automatically, and require minimal user interaction. These are features, not bugs, but they create a threat surface that maps cleanly to two CWE categories the researchers identify: CWE-494 (Download of Code Without Integrity Check) and CWE-345 (Insufficient Verification of Data Authenticity).

The attack surface lives in the trust model governing ClickOnce manifests. A deployment manifest references an application manifest, which in turn describes the payload to be installed. If a threat actor controls the hosting infrastructure, whether a compromised web server or a rogue network file share, they can serve a malicious application manifest that ClickOnce will process with the same trust it extends to legitimate software. The

absence of strong cryptographic integrity verification at the deployment layer means that a user clicking a ClickOnce link is, in many configurations, trusting the delivery channel rather than the payload itself.

The MITRE ATT&CK techniques cited in this research tell a coherent attack story. T1189 (Drive-by Compromise) and T1204.002 (Malicious File execution) describe the initial delivery; T1105 (Ingress Tool Transfer) covers payload staging; T1036 (Masquerading) explains how the legitimate ClickOnce wrapper lends credibility to malicious content; T1072 (Software Deployment Tools) and T1195 (Supply Chain Compromise) reflect the broader category of abusing trusted deployment infrastructure; T1071.001 (Application Layer Protocol: Web Protocols) describes C2 or payload delivery over HTTP/HTTPS; and T1547 (Boot or Logon Autostart Execution) points to persistence mechanisms that a fully deployed ClickOnce application can establish.

Critically, this is a Part 1 report. CrowdStrike explicitly defers specific exploitation techniques and defender guidance to Part 2. What Part 1 establishes is the foundational threat model: a widely deployed, Microsoft-supported deployment mechanism that security teams have largely treated as benign is structurally capable of serving as a full malware delivery pipeline. Organizations that have not audited ClickOnce usage, whitelisting policies, or application manifest trust configurations should treat this as an advance warning, not a completed threat assessment.

## Action Checklist

1. Step 1: Assess exposure, audit your environment for ClickOnce application deployments; query installed applications on endpoints for .application or .appref-ms file associations, which are ClickOnce deployment artifacts
2. Step 2: Review application allowlisting controls, verify that your application control policies (per NIST CM controls and CIS Controls v8 2.1, 2.3) treat ClickOnce-deployed applications as a distinct category requiring review, not an implicit exception
3. Step 3: Audit network file share and web server access, identify any internal or externally accessible paths currently serving ClickOnce manifests; apply NIST AC-4 (Information Flow Enforcement) to restrict who can host or modify deployment manifests
4. Step 4: Review code-signing and manifest integrity requirements, determine whether your ClickOnce deployments enforce Authenticode signing on both the deployment manifest and the application manifest; unsigned or self-signed deployments represent the highest-risk configurations
5. Step 5: Brief security operations, alert your SOC to the T1072 (Software Deployment Tools) and T1204.002 (Malicious File) technique chain; ensure EDR rules are reviewed for ClickOnce process chains before Part 2 of the CrowdStrike series publishes additional exploitation detail
6. Step 6: Monitor for Part 2, track the CrowdStrike blog for the follow-up installment, which will contain specific exploitation techniques and detection/defense guidance; assign ownership for intake and action on that release

## IR / Forensic Enrichment

| | |
|---|---|
| **Triage Priority** | STANDARD |

| | |
|---|---|
| **Escalation Criteria** | Escalate to urgent if active `dfsvc.exe` child process chains are detected on any endpoint, if `.application` manifests are found on internal file shares with no corresponding authorized deployment record, or if the organization hosts externally accessible ClickOnce distribution points serving unsigned or self-signed manifests — any of these conditions indicates active or imminent exploitation risk warranting immediate containment and IR lead engagement. |
| **Recovery Notes** | Once unauthorized ClickOnce deployments are confirmed and contained, verify removal of all malicious `.application`, `.appref-ms`, and `.deploy` artifacts from `%LOCALAPPDATA%\Apps\2.0\` on affected endpoints and from any hosting file shares or web directories. Reimage endpoints where ClickOnce-delivered payloads executed with confirmed post-exploitation activity (lateral movement, credential access, or persistence) rather than attempting surgical cleanup, given ClickOnce's per-user install path which can be abused to maintain persistence outside standard program directories. Monitor `dfsvc.exe` process creation events and outbound connections from the ClickOnce deployment cache path for a minimum of 30 days post-remediation to detect reinfection attempts via previously distributed manifest URLs that may still be cached or bookmarked by end users. |
| **Forensic Artifacts** | ClickOnce deployment cache directory: `%LOCALAPPDATA%\Apps\2.0\` — contains extracted application binaries, `.application` deployment manifests, and `.manifest` application manifests with timestamps that establish when a malicious payload was first deployed and from which source URL (source URL is embedded in the deployment manifest XML under the `deploymentProvider` element) \| Windows Event Log — Microsoft-Windows-ClickOnce/Operational (`Applications and Services Logs\Microsoft\Windows\ClickOnce\Operational`) — logs deployment activations, manifest download URLs, and trust decisions including whether the user accepted an untrusted publisher prompt, providing a timeline of ClickOnce execution events \| Sysmon Event ID 1 (Process Create) and Event ID 3 (Network Connection) for `dfsvc.exe` — captures the full process chain initiated by ClickOnce activation, including any suspicious child processes and outbound network connections to attacker-controlled manifest-hosting infrastructure made during the deployment fetch \| IIS or web server access logs for the manifest-hosting server (`%SystemDrive%\inetpub\logs\LogFiles\W3SVC*\*.log`) — GET requests to `.application` URI paths reveal which client IPs fetched the malicious manifest, enabling victim scoping and timeline reconstruction of the delivery campaign \| Registry key `HKCU:\Software\Microsoft\Windows\CurrentVersion\Uninstall\` entries with `UninstallString` values referencing `rundll32.exe dfshim.dll` — these are left by every ClickOnce application registered in the current user's context and provide an inventory of all ClickOnce-installed applications including malicious ones, with install date metadata |

### Per-Action IR Details

**Step 1: Assess exposure — audit your environment for ClickOnce application deployments; query installed applications on endpoints for .application or .appref-ms file associations, which are ClickOnce deployment artifacts**

**NIST Phase:** Preparation

**Reference:** NIST 800-61r3 §2 — Preparation: establishing visibility into deployment artifacts before an incident occurs

**Controls:** CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory), CIS 2.1 (Establish and Maintain a Software Inventory)

**Compensating:** Run the following on all Windows endpoints via PowerShell remoting or a GPO-deployed script: `Get-ChildItem -Path $env:LOCALAPPDATA\Apps\2.0 -Recurse -Include *.application,*.appref-ms | Select-Object FullName,LastWriteTime` to enumerate ClickOnce artifacts in the per-user deployment directory. Cross-reference against `HKCU:\Software\Microsoft\Windows\CurrentVersion\Uninstall` for ClickOnce-installed app entries. Export results to a central CSV for manual review by the two-person team.

**Evidence:** This step is a passive audit and does not alter live state. However, before touching any endpoint flagged as potentially compromised during the audit, capture: contents of `%LOCALAPPDATA%\Apps\2.0\` (ClickOnce deployment cache, including .application manifests and extracted binaries), `%TEMP%` and `%APPDATA%\Microsoft\Windows\Start Menu\Programs\` for .appref-ms shortcuts placed by malicious deployments, and the registry key `HKCU:\Software\Microsoft\Windows\CurrentVersion\Uninstall` to identify any ClickOnce-registered application names and install timestamps. Preserve file metadata (creation, modification, access times) before enumeration tools alter them.

**Step 2: Review application allowlisting controls — verify that your application control policies (per NIST CM controls and CIS 2.1, CIS 2.3) treat ClickOnce-deployed applications as a distinct category requiring review, not an implicit exception**

**NIST Phase:** Preparation

**Reference:** NIST 800-61r3 §2 — Preparation: hardening and policy validation to reduce attack surface prior to exploitation

**Controls:** CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.3 (Address Unauthorized Software), CIS 2.2 (Ensure Authorized Software is Currently Supported)

**Compensating:** Use AppLocker (built into Windows Pro/Enterprise) to create a rule specifically targeting the ClickOnce deployment path: create a Path rule blocking execution from `%LOCALAPPDATA%\Apps\2.0\` for all users except explicitly approved publisher-signed applications. Export current AppLocker policy via `Get-AppLockerPolicy -Effective | Export-Clixml applocker_current.xml` for baseline documentation. For endpoints without AppLocker, use Software Restriction Policies (SRP) to set a Disallowed rule on the `%LOCALAPPDATA%\Apps\2.0` path. Document any explicit exceptions with business justification.

**Evidence:** This is a policy review and configuration step, not an action that alters live endpoint state. No volatile capture is required before this step. Document the pre-change policy state by exporting existing AppLocker or SRP rules before any modifications. Record which ClickOnce-deployed applications were previously permitted implicitly, as this list is forensically relevant if malicious deployments are later discovered to have executed under the prior permissive policy.

**Step 3: Audit network file share and web server access — identify any internal or externally accessible paths currently serving ClickOnce manifests; apply NIST AC-4 (Information Flow Enforcement) to restrict who can host or modify deployment manifests**

**NIST Phase:** Preparation

**Reference:** NIST 800-61r3 §2 — Preparation: reducing attack surface by controlling distribution infrastructure for ClickOnce manifests

**Controls:** NIST AC-4 (Information Flow Enforcement), CIS 4.4 (Implement and Manage a Firewall on Servers), CIS 4.2 (Establish and Maintain a Secure Configuration Process for Network Infrastructure)

**Compensating:** On IIS or Apache hosts, audit the directory listings for `.application` and `.manifest` files: `find /var/www -name '*.application' -o -name '*.manifest' 2>/dev/null` (Linux) or `dir /s /b *.application *.manifest` on Windows web roots. For SMB shares, run `Get-SmbShare | ForEach-Object { Get-ChildItem -Path $_.Path -Recurse -Include *.application,*.appref-ms -ErrorAction SilentlyContinue }`. Review IIS access logs (`%SystemDrive%\inetpub\logs\LogFiles\`) for GET requests to `.application` paths — filter for external source IPs accessing manifest URLs. Restrict write access to manifest-hosting directories using NTFS ACLs and verify with `icacls `.

**Evidence:** Before restricting or modifying any file share or web directory ACL, capture: current IIS or web server access logs (`%SystemDrive%\inetpub\logs\LogFiles\W3SVC*\*.log`) covering the past 90 days, filtering for requests to `.application` and `.manifest` URIs; current NTFS ACL state of all identified manifest-hosting directories via `icacls > acl_baseline.txt`; SMB share permission exports via `Get-SmbShareAccess`; and any `.application` or `.deploy` manifest files currently present on the shares (hash them with `Get-FileHash` before any changes are made). These establish a baseline and may reveal unauthorized manifest modifications or external access patterns consistent with ClickOnce-based delivery staging.

**Step 4: Review code-signing and manifest integrity requirements — determine whether your ClickOnce deployments enforce Authenticode signing on both the deployment manifest and the application manifest; unsigned or self-signed deployments represent the highest-risk configurations**

**NIST Phase:** Preparation

**Reference:** NIST 800-61r3 §2 — Preparation: identifying trust-chain gaps in ClickOnce deployment infrastructure that enable malicious payload substitution

**Controls:** CIS 4.6 (Securely Manage Enterprise Assets and Software)

**Compensating:** Use `mage.exe` (included in the .NET SDK, freely available) to inspect existing ClickOnce manifests: `mage -verify .application` will report whether the deployment manifest carries a valid Authenticode signature from a trusted publisher. Alternatively, use `sigcheck.exe` from Sysinternals (free) against all `.application` files found in Step 1: `sigcheck -a -nobanner *.application`. Enumerate self-signed certificates used for ClickOnce via `Get-ChildItem Cert:\CurrentUser\My | Where-Object { $_.Subject -eq $_.Issuer }` and cross-reference with known legitimate internal CA issuers. Flag any deployment signed with a self-signed or expired certificate for immediate review.

**Evidence:** This is a configuration review step and does not alter live endpoint state. Before any remediation that revokes or replaces signing certificates, capture: the certificate thumbprints and full certificate chains for all ClickOnce signing certificates currently in use (`Get-ChildItem Cert:\LocalMachine\My | Format-List Thumbprint,Subject,Issuer,NotAfter`); existing `.application` manifest files with their current hashes (`Get-FileHash`); and the ClickOnce trust prompt level configured in the registry at `HKLM\SOFTWARE\Microsoft\.NETFramework\Security\TrustManager\PromptingLevel`, which controls whether unsigned or self-signed deployments are permitted to execute silently — this value is a critical forensic indicator of the trust boundary that an attacker would target.

**Step 5: Brief security operations — alert your SOC to the T1072 (Software Deployment Tools) and T1204.002 (Malicious File) technique chain; ensure EDR rules are reviewed for ClickOnce process chains before Part 2 of the CrowdStrike series publishes additional exploitation detail**

**NIST Phase:** Detection Analysis

**Reference:** NIST 800-61r3 §3.2 — Detection and Analysis: equipping analysts with technique-specific detection logic for ClickOnce-based malware delivery chains

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

**Compensating:** Deploy or update Sysmon configuration to capture the ClickOnce process chain: `dfsvc.exe` (the ClickOnce deployment service) spawning child processes is a high-fidelity detection signal. Add a Sysmon rule targeting Event ID 1 (Process Create) where ParentImage contains `dfsvc.exe` and target CommandLine spawns `cmd.exe`, `powershell.exe`, `wscript.exe`, `mshta.exe`, or `rundll32.exe`. A matching Sigma rule: `title: ClickOnce dfsvc.exe Spawning Suspicious Child Process; logsource: product: windows, category: process_creation; detection: selection: ParentImage|endswith: '\dfsvc.exe', Image|endswith: ['\cmd.exe','\powershell.exe','\wscript.exe','\mshta.exe','\rundll32.exe']; condition: selection`. Forward Sysmon Event IDs 1, 3, and 7 (Process Create, Network Connection, Image Load) with parent `dfsvc.exe` to a central log store for analyst review.

**Evidence:** Before modifying any EDR rules or Sysmon configuration, capture the current detection baseline: export existing EDR policy rules and any Sysmon configuration XML (`sysmon -c` outputs current config) so that gaps are documented prior to the update. This step does not alter live endpoint state, but if an analyst identifies an active `dfsvc.exe` process chain on a live host during the briefing review, volatile evidence must be captured immediately before any containment action: acquire a full memory image of the host using WinPmem or Magnet RAM Capture, capture `Get-Process | Where-Object { $_.Name -eq 'dfsvc' } | Select-Object *` output, and run `Get-NetTCPConnection -OwningProcess ` to capture any active network connections established by the ClickOnce process before the session is terminated.

**Step 6: Monitor for Part 2 — track the CrowdStrike blog for the follow-up installment, which will contain specific exploitation techniques and detection/defense guidance; assign ownership for intake and action on that release**

**NIST Phase:** Post Incident

**Reference:** NIST 800-61r3 §4 — Post-Incident Activity: integrating emerging threat intelligence into defensive posture and updating IR capabilities ahead of published exploitation detail

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

**Compensating:** Assign one team member to set an RSS or email alert on `https://www.crowdstrike.com/blog/` filtered for 'ClickOnce' as the monitoring mechanism. Create a tracking ticket (GitHub Issues, Jira, or a shared spreadsheet for small teams) with the assigned owner, expected intake SLA (recommend 24 hours from publication), and a checklist: (1) extract any new IOCs or detection signatures, (2) map new technique details to existing Sysmon/EDR rules from Step 5, (3) update AppLocker or SRP rules from Step 2 if new delivery paths are disclosed, (4) redistribute to the team within 48 hours of publication. Document the assignment and SLA in writing so accountability is clear if the follow-up publishes exploitation details during an ongoing incident.

**Evidence:** This is a planning and intelligence-tracking step with no live system interaction. No volatile capture is required. However, preserve the current state of all artifacts gathered in Steps 1–5 (inventory exports, manifest hashes, ACL baselines, Sysmon configuration snapshots, and EDR rule exports) as a dated forensic baseline. When Part 2 publishes, compare any disclosed IOCs — such as known-malicious `.application` manifest hashes, C2 domains contacted by ClickOnce-delivered payloads, or specific ClickOnce deployment URLs used in campaigns — against this baseline to determine retroactive exposure.

## Detection Guidance

Detection at this stage is necessarily behavioral, as no CVE and no specific IOCs have been published in Part 1 of this series.

Process chain monitoring: ClickOnce applications execute through dfsvc.exe (the ClickOnce deployment service host). Hunt for dfsvc.exe spawning unexpected child processes, particularly scripting interpreters (wscript.exe, mshta.exe, powershell.exe) or command shells. This process lineage is unusual in legitimate ClickOnce deployments. Map to NIST SI-4 (System Monitoring) and align with D3FEND countermeasure D3-SFA (System File Analysis).

Manifest source monitoring: Log and alert on .application and .appref-ms file downloads, particularly from external web servers or unusual internal hosts. Cross-reference against your authorized ClickOnce application inventory. NIST AU-2 (Event Logging) should capture file-type download events at the proxy or endpoint.

Network file share auditing: Monitor for access to UNC paths serving .application files. Unexpected hosts appearing as ClickOnce deployment sources warrant immediate investigation. NIST AU-6 (Audit Record Review, Analysis, and Reporting) should drive regular review of these events.

Signing anomalies: Where your environment has previously enforced signed ClickOnce deployments, alert on any manifest that arrives without a valid Authenticode signature or with a certificate not in your trusted publisher list. D3FEND countermeasure D3-ACA (Active Certificate Analysis) is directly applicable here.

User execution patterns: T1204.002 depends on user interaction. Correlate ClickOnce execution events against phishing indicators, inbound email links, or web proxy logs showing navigation to domains not previously associated with ClickOnce delivery. Anomalous first-seen domains serving .application files are high-fidelity hunt pivots.

Account behavior post-execution: T1547 (persistence) and T1105 (ingress tool transfer) suggest looking for new autorun registry entries or scheduled tasks created shortly after dfsvc.exe activity. Apply NIST AC-6 (Least Privilege) and D3FEND D3-LAM (Local Account Monitoring) to surface privilege escalation attempts following initial ClickOnce execution.

## Indicators of Compromise

| Type | Value | Context | Confidence |
|---|---|---|---|
| TOOL | `dfsvc.exe` | dfsvc.exe (ClickOnce Deployment Service Host) leveraged as the execution host for malicious ClickOnce application manifests served from attacker-controlled web servers or network file shares to install payloads without administrator privileges | MEDIUM |
| URL | `Pending — refer to CrowdStrike blog Part 2 for published indicators` | Part 1 of the CrowdStrike series establishes the threat model only; specific payload hashes, C2 domains, and malicious manifest URLs are expected in the follow-up installment | LOW |

## Framework Mappings

### MITRE-ATTACK

- **T1105** — Ingress Tool Transfer
- **T1036** — Masquerading
- **T1072** — Software Deployment Tools
- **T1071.001** — Web Protocols
- **T1204.002** — Malicious File
- **T1547** — Boot or Logon Autostart Execution
- **T1195** — Supply Chain Compromise
- **T1189** — Drive-by Compromise

### NIST-800-53R5

- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **CM-7** — Least Functionality
- **SA-9** — External System Services
- **SR-2** — Supply Chain Risk Management Plan
- **SR-3** — Supply Chain Controls and Processes
- **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
- **8.2** — Collect Audit Logs

**ISO-27001-2022**

- **A.8.8** — Management of technical vulnerabilities

**NIST-CSF-2**

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

## MITRE ATT&CK Mapping

| Technique ID | Technique Name | Tactic |
|---|---|---|
| T1105 | Ingress Tool Transfer | Command-And-Control |
| T1036 | Masquerading | Defense-Evasion |
| T1072 | Software Deployment Tools | Execution |
| T1071.001 | Web Protocols | Command-And-Control |
| T1204.002 | Malicious File | Execution |
| T1547 | Boot or Logon Autostart Execution | Persistence |
| T1195 | Supply Chain Compromise | Initial-Access |
| T1189 | Drive-by Compromise | 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/why-small-businesses-choose-... | T3 |
| | https://www.crowdstrike.com/en-us/blog/how-the-infrastructure-inves... | T3 |
| | https://www.crowdstrike.com/en-us/blog/reasons-why-nonprofits-are-t... | T3 |
| **ClickOnce Deployment and Security - Visual Studio (Windows)** | https://learn.microsoft.com/en-us/visualstudio/deployment/clickonce... | T1 |

## DISCLAIMER

This intelligence report is produced by Tech Jacks Solutions Security Command Center (SCC) for informational purposes only. It does not constitute professional security advice, legal counsel, or an incident response engagement. The information herein is derived from publicly available sources and AI-assisted analysis; while every effort is made to ensure accuracy, Tech Jacks Solutions makes no warranties regarding completeness or timeliness. Organizations should conduct their own validation and consult qualified security professionals before taking action based on this report. Tech Jacks Solutions is not liable for any damages resulting from the use of this information.

Generated 2026-06-21 18:39 UTC by TJS Security Command Center