INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-06 18:18 UTC

UNC4736 (Labyrinth Chollima) Executes $280M+ Drift Protocol Heist via Six-Month Social Engineering Campaign

THREAT CAMPAIGN | **HIGH** | CVSS 9.5

SCC Item ID	SCC-CAM-2026-0152
Type	Threat Campaign
Severity	HIGH
CVSS Base Score	9.5
Affected Products	Drift Protocol (Solana-based DeFi platform), VSCode, Cursor IDE, Apple TestFlight
Published	2026-04-06T12:35:03
Discovery Source	Rss

Executive Summary

UNC4736 (Labyrinth Chollima), a North Korean state-sponsored threat actor, stole approximately $280-285 million from Drift Protocol, a Solana-based decentralized finance platform, after a six-month operation that combined in-person social engineering at crypto industry conferences with malware delivered through poisoned developer tooling and a fraudulent mobile application. Attackers gained administrative control of Drift's Security Council and drained user funds in roughly 12 minutes. Any organization operating DeFi infrastructure, employing developers active in the cryptocurrency space, or holding significant on-chain assets faces equivalent risk from DPRK-affiliated actors who now conduct sustained HUMINT-style operations rather than opportunistic phishing.

Technical Analysis

UNC4736 executed a multi-phase intrusion against Drift Protocol with no discrete CVE assigned, this is a campaign-level event spanning supply chain compromise and social engineering. No authoritative CVSS scoring exists; a 9.5 base score appears in source data but is unverified by NVD or the CVSS SIG and should be treated as an estimate. Relevant CWEs: CWE-94 (Code Injection via poisoned repository targeting VSCode and Cursor IDE users), CWE-284 (Improper Access Control, Security Council administrative takeover), CWE-346 (Origin Validation Error, fraudulent Apple TestFlight application trusted by targets), CWE-506 (Embedded Malicious Code in developer tooling). MITRE ATT&CK techniques involved include T1195.001 (Supply Chain Compromise: Compromise Software Dependencies), T1566/T1566.003 (Phishing/Spearphishing via social engineering and fraudulent apps), T1204.002 (Malicious File execution), T1036 (Masquerading,

non-Korean intermediaries posing as quant firm representatives), T1078 (Valid Accounts, Security Council compromise), T1547 (Boot/Logon Autostart Persistence), T1552 (Unsecured Credentials), T1562 (Impair Defenses), T1070.004 (File Deletion), T1560 (Archive Collected Data), T1591.004 (Gather Victim Org Information: Identify Roles), T1588.001 (Obtain Capabilities: Malware), and T1657 (Financial Theft). The attack chain: prolonged conference-based relationship cultivation with Drift contributors, malware delivery via a poisoned code repository presented to VSCode and Cursor IDE users, a fraudulent TestFlight app as a secondary vector, credential and access harvesting, and a 12-minute fund drain once Security Council control was achieved.

Action Checklist

1. Step 1: Containment, If your organization operates DeFi protocols or multi-sig governance structures, immediately audit current Security Council or multisig signers for unauthorized additions or credential exposure. Revoke any signer access added in the past six months without a documented, verified approval chain. Suspend external contributor access to administrative roles pending re-verification of identities through out-of-band channels.
2. Step 2: Detection, Audit developer workstations for VSCode and Cursor IDE extension installations from unverified publishers, particularly any installed in the past six months. Search endpoint telemetry for processes spawned by IDE extension hosts that initiate outbound network connections, write to startup locations (MITRE T1547), or access credential stores (T1552). Review Apple TestFlight app installations across developer devices for applications not approved through your software inventory process. Cross-reference conference attendance records against personnel who have administrative access to on-chain governance mechanisms.
3. Step 3: Eradication, Remove any unverified IDE extensions and re-image affected developer workstations rather than attempting surgical cleanup given the persistence mechanisms involved (T1547, T1562). Rotate all credentials and private keys accessible from potentially compromised developer environments. Revoke and reissue multisig or Security Council credentials through a verified, documented process. Remove any unauthorized TestFlight applications and revoke associated device trust.
4. Step 4: Recovery, Verify Security Council or multisig composition matches your approved signer list through on-chain audit and out-of-band identity confirmation for each signer. Monitor on-chain governance actions and fund movements in real time for at least 30 days post-remediation. Validate that no persistence mechanisms (scheduled tasks, autostart entries, browser credential stores) remain on re-imaged or cleaned systems before restoring administrative access.
5. Step 5: Post-Incident, This campaign exposes three specific control gaps: (1) absence of identity verification for individuals granted access to governance roles, relying instead on cultivated trust rather than documented processes; (2) no separation between developer workstation environments and administrative key material; (3) no behavioral monitoring on IDE processes or extension activity. Implement hardware security keys for all governance signers, enforce developer workstation isolation from key management systems, and establish a vetting protocol for any external individual granted contributor or governance access, including in-person verification for high-trust roles. Note: this is an operational security and governance failure, not a patchable software defect.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate immediately to executive leadership, legal counsel, and relevant regulators (FinCEN SAR filing obligation if U.S.-nexus funds exceed $5,000; OFAC nexus given DPRK state sponsorship of UNC4736) if any on-chain fund movement is detected post-containment, if a Security Council signer cannot be verified through out-of-band channels within 2 hours, or if forensic analysis reveals the compromise window extends beyond six months or touches systems outside the developer workstation environment.
Recovery Notes	Reconstitute the Drift Security Council only after every signer has been re-verified through a documented out-of-band process and new keys have been generated exclusively on air-gapped or verified-clean hardware — any signing key that was accessible from a VSCode/Cursor-equipped developer workstation during the six-month window must be treated as fully compromised. Monitor all on-chain governance proposals and fund movements via automated alerting (Helius, Triton, or custom Solana RPC webhook) for a minimum of 30 days post-recovery, with manual daily review of the Security Council transaction log by a designated team member. Given UNC4736's documented patience (six-month campaign) and state-level resources, treat any anomalous developer contractor outreach or conference networking from individuals seeking governance access as a potential re-engagement attempt and apply the full vetting protocol before any access is considered.
Forensic Artifacts	VSCode/Cursor extension directories (~/.vscode/extensions/ or %USERPROFILE%\.vscode\extensions\) — malicious UNC4736 extensions would contain obfuscated JavaScript with beacon logic; preserve full directory with file timestamps and compute SHA-256 hashes of all .vsix and extension manifest files before removal | Solana RPC transaction history for the Drift Security Council governance program address — 'getSignaturesForAddress' covering the full six-month window will show unauthorized signer additions, proposal submissions, and the fund drain transactions with exact timestamps and initiating wallet addresses | macOS Unified Log archive or Windows Sysmon Event ID 3 (Network Connection) logs filtered on Electron/extensionHost processes — these capture the C2 beacon traffic initiated by the poisoned IDE extension, including destination IPs, ports, and connection frequency that characterize UNC4736's implant communication pattern | Apple TestFlight application binary and associated Info.plist from /private/var/containers/Bundle/Application/ on affected iOS/macOS devices — the fraudulent TestFlight app used in this campaign would contain entitlements or embedded URLs pointing to UNC4736 infrastructure not present in legitimate developer tooling | Developer workstation browser credential stores (Chrome Login Data SQLite DB, macOS Keychain entries for 'solana' or 'ledger') — UNC4736 tooling consistent with Lazarus cluster operations has targeted browser-stored credentials and local key material; these stores establish whether private key or seed phrase exfiltration occurred beyond the on-chain evidence

Per-Action IR Details

Step 1: Containment — If your organization operates DeFi protocols or multi-sig governance structures, immediately audit current Security Council or multisig signers for unauthorized additions or credential exposure. Revoke any signer access added in the past six months without a documented, verified approval chain. Suspend external contributor access to administrative roles pending re-verification of identities through out-of-band channels.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy: isolate affected systems and revoke unauthorized access before further damage occurs; aligns with CSF RS.MA-01 (execute IR plan in coordination with relevant third parties, including on-chain governance participants)

Controls: NIST IR-4 (Incident Handling), NIST AC-2 (Account Management) — revoke unauthorized signer accounts added without documented approval, NIST AC-6 (Least Privilege) — limit Security Council membership to verified, need-to-know signers only, CIS 5.1 (Establish and Maintain an Inventory of Accounts) — verify all multisig signer accounts against approved inventory, CIS 6.2 (Establish an Access Revoking Process) — execute documented process to revoke unauthorized signer access immediately

Compensating: For a 2-person team with no enterprise IAM tooling: (1) Pull the on-chain signer list directly via Solana CLI — 'solana account ' or use Drift's on-chain governance explorer — and diff against your last documented approval record. (2) For each signer added in the past six months, require video call verification with a second known contact using a pre-established out-of-band channel (Signal with safety numbers verified). (3) Immediately submit an on-chain transaction to remove any unverified signers; do not wait for identity confirmation before removal if the approval chain is absent.

Evidence: Before revoking access, capture: (1) Full on-chain transaction history for the Drift Security Council program address covering the six-month window — export via Solana RPC 'getSignaturesForAddress' and preserve as immutable evidence. (2) Timestamps and wallet addresses of all signer additions, including the proposing wallet and approval signatures. (3) Any GitHub, Telegram, or Discord communications used to onboard the suspected compromised signer — these establish the social engineering timeline. (4) Developer workstation SSH key exports and GPG key fingerprints used to authenticate to governance tooling.

Step 2: Detection — Audit developer workstations for VSCode and Cursor IDE extension installations from unverified publishers, particularly any installed in the past six months. Search endpoint telemetry for processes spawned by IDE extension hosts that initiate outbound network connections, write to startup locations (MITRE T1547), or access credential stores (T1552). Review Apple TestFlight app installations across developer devices for applications not approved through your software inventory process. Cross-reference conference attendance records against personnel who have administrative access to on-chain governance mechanisms.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis: correlate endpoint telemetry, extension installation logs, and identity records to reconstruct the six-month UNC4736 access timeline; aligns with CSF DE.AE-03 (correlate information from multiple sources) and DE.AE-07 (integrate threat intelligence into analysis)

Controls: NIST SI-4 (System Monitoring) — monitor IDE extension host processes for anomalous outbound connections and credential access, NIST AU-6 (Audit Record Review, Analysis, and Reporting) — review VSCode/Cursor extension installation logs and macOS Unified Log entries, NIST AU-2 (Event Logging) — ensure extension installation events and process creation are captured in endpoint logs, NIST IR-5 (Incident Monitoring) — track and document all detected indicators across affected developer workstations, CIS 2.1 (Establish and Maintain a Software Inventory) — identify all VSCode and Cursor IDE extensions against approved software inventory, CIS 8.2 (Collect Audit Logs) — confirm audit logging was enabled on developer endpoints before and during the attack window

Compensating: Deploy Sysmon (config: SwiftOnSecurity baseline minimum) on Windows developer workstations and enable Event ID 1 (Process Create) and Event ID 3 (Network Connect). Query with: 'Get-WinEvent -LogName "Microsoft-Windows-Sysmon/Operational" | Where {$_.Id -eq 3 -and $_.Message -match "extensionHost"}' to surface outbound connections from VSCode/Cursor extension hosts. On macOS, run 'log show --predicate "process == 'Electron'" --last 180d' and filter for network activity from the extension host PID. For TestFlight: run 'system_profiler SPApplicationsDataType | grep -i testflight' and cross-reference against your approved software inventory. Use osquery — 'SELECT * FROM apps WHERE name LIKE "%testflight%";' — and 'SELECT * FROM file WHERE path LIKE "/Users/%/.vscode/extensions/%" AND ctime > (strftime("%s","now") - 15552000);' to enumerate extensions installed in the past six months.

Evidence: Before remediating, preserve: (1) Full VSCode/Cursor extension directories — '%USERPROFILE%\.vscode\extensions\' (Windows) or '~/.vscode/extensions/' (macOS/Linux) — tar/zip with timestamps intact. (2) Sysmon Event ID 1 and Event ID 3 logs filtered on 'extensionHost.exe' or Electron-based processes showing C2 beacon patterns (periodic outbound to non-CDN IPs). (3) macOS Unified Log entries from the suspected compromise window: 'log collect --last 180d --output ~/ir_evidence/unified.logarchive'. (4) Apple TestFlight app binary and associated plist from '/private/var/containers/Bundle/Application/' for the unauthorized app. (5) Solana wallet private key file locations (e.g., '~/.config/solana/id.json') — do NOT copy the key material, but document whether

key files were accessible from the compromised session.

Step 3: Eradication — Remove any unverified IDE extensions and re-image affected developer workstations rather than attempting surgical cleanup given the persistence mechanisms involved (T1547, T1562). Rotate all credentials and private keys accessible from potentially compromised developer environments. Revoke and reissue multisig or Security Council credentials through a verified, documented process. Remove any unauthorized TestFlight applications and revoke associated device trust.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication: eliminate all threat actor footholds including T1547 autostart persistence and T1562 defense evasion artifacts planted by UNC4736's poisoned IDE extensions; full re-image is required given the steganographic or fileless persistence capabilities associated with Lazarus-cluster tooling

Controls: NIST SI-2 (Flaw Remediation) — remove malicious extensions and re-image systems to eliminate all threat actor artifacts, NIST SI-3 (Malicious Code Protection) — scan re-imaged systems with updated signatures before restoring to production, NIST SI-7 (Software, Firmware, and Information Integrity) — verify integrity of development tooling and extension manifests post-eradication, NIST CM-4 (Impact Analyses) — assess impact of compromised private keys before rotating to understand blast radius, CIS 2.3 (Address Unauthorized Software) — remove all unauthorized VSCode/Cursor extensions and TestFlight applications per documented process, CIS 4.6 (Securely Manage Enterprise Assets and Software) — enforce clean OS image deployment from verified golden image

Compensating: For re-imaging without enterprise MDM: (1) Boot from a verified OS installer (USB created via Apple Configurator 2 or Windows Media Creation Tool on an uncompromised machine). (2) Before wiping, run YARA scan using published UNC4736/Lazarus YARA rules from CISA advisories (AA22-108A and related) against the extension directories: 'yara -r lazarus_rules.yar ~/.vscode/extensions/'. (3) For Solana key rotation, generate new keypairs on an air-gapped machine: 'solana-keygen new --outfile ~/new-wallet.json' — never generate replacement keys on a potentially compromised host. (4) For TestFlight removal on iOS, use 'cfgutil' (Apple Configurator) or MDM profile removal; if no MDM, perform factory reset of affected iOS devices. (5) Document each rotation event with timestamp, old key fingerprint (public only), and new key fingerprint for the incident record.

Evidence: Capture before re-imaging: (1) Full disk image of affected developer workstations using dd or FTK Imager — UNC4736 tooling has included implants that survive extension removal, so surgical cleanup evidence will be incomplete without a full image. (2) Registry export of 'HKCU\Software\Microsoft\Windows\CurrentVersion\Run' and 'HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon' for T1547 autostart artifacts. (3) macOS LaunchAgents and LaunchDaemons: 'ls -la ~/Library/LaunchAgents/ /Library/LaunchAgents/ /Library/LaunchDaemons/' — UNC4736 has used LaunchAgent persistence in prior macOS campaigns. (4) Network connection state at time of discovery: 'netstat -an' / 'ss -tulnp' output saved to file. (5) Hash (SHA-256) of all installed extension .vsix packages for later threat intelligence sharing.

Step 4: Recovery — Verify Security Council or multisig composition matches your approved signer list through on-chain audit and out-of-band identity confirmation for each signer. Monitor on-chain governance actions and fund movements in real time for at least 30 days post-remediation. Validate that no persistence mechanisms (scheduled tasks, autostart entries, browser credential stores) remain on re-imaged or cleaned systems before restoring administrative access.

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery: restore Drift Security Council to a verified-clean composition, confirm eradication of UNC4736 persistence (T1547 autostart, browser credential stores per T1555), and establish a 30-day heightened monitoring window before returning governance keys to production use; aligns with CSF RC functions (execute recovery plan, verify integrity, communicate)

Controls: NIST IR-4 (Incident Handling) — execute recovery phase of incident handling plan including system restoration and verification, NIST SI-6 (Security and Privacy Function Verification) — verify correct operation of governance mechanisms and signing processes post-recovery, NIST SI-7 (Software, Firmware, and Information Integrity) — validate integrity of re-imaged workstations before restoring administrative key access, NIST AU-6 (Audit Record Review, Analysis, and Reporting) — review on-chain governance transaction logs and endpoint audit records during 30-day monitoring window, CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory) — verify

re-imaged workstations are documented in asset inventory before returning to production, CIS 5.1 (Establish and Maintain an Inventory of Accounts) — confirm final Security Council signer inventory matches approved list before restoring governance access

Compensating: For a 2-person team: (1) On-chain signer verification — use 'solana program show ' and manually verify each remaining signer public key against your written approval record, confirmed via video call with each signer's known contact. (2) Set up a free Solana on-chain monitoring alert using Helius or Triton webhooks (free tier) to trigger on any Security Council proposal or fund transfer above a defined threshold. (3) On re-imaged workstations, run Sysinternals Autoruns (Windows) or 'launchctl list' + 'crontab -l' (macOS) and compare output against a clean baseline before granting key access. (4) Check browser credential stores manually: Chrome — 'sqlite3 ~/Library/Application\ Support/Google/Chrome/Default/Login\ Data ".headers on" "SELECT origin_url, username_value FROM logins;"' — to confirm no stored governance credentials persist.

Evidence: Before restoring governance access, document: (1) Final on-chain state of Security Council — export signer list and threshold configuration as a timestamped record via Solana RPC. (2) Autoruns or LaunchAgent baseline comparison output confirming no T1547 persistence artifacts remain on re-imaged systems. (3) Signed attestation (email or chat log) from each re-verified Security Council signer confirming identity through out-of-band channel. (4) Screenshot or export of new Solana wallet public key fingerprints enrolled in the reconstituted multisig, for inclusion in the post-incident report.

Step 5: Post-Incident — This campaign exposes three specific control gaps: (1) absence of identity verification for individuals granted access to governance roles, relying instead on cultivated trust rather than documented processes; (2) no separation between developer workstation environments and administrative key material; (3) no behavioral monitoring on IDE processes or extension activity. Implement hardware security keys for all governance signers, enforce developer workstation isolation from key management systems, and establish a vetting protocol for any external individual granted contributor or governance access — including in-person verification for high-trust roles.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: conduct lessons-learned review documenting the three control gaps UNC4736 exploited — identity trust without verification, key material co-location with developer tooling, and absence of IDE behavioral monitoring — and update IR plan, detection rules, and onboarding vetting procedures accordingly; aligns with CSF GV and ID functions (update policies, improve detection, share intelligence)

Controls: NIST IR-4 (Incident Handling) — update incident handling procedures to include vetting requirements for governance role onboarding, NIST IR-8 (Incident Response Plan) — revise IR plan to incorporate DeFi-specific governance attack scenarios based on this campaign, NIST IR-2 (Incident Response Training) — train developer and governance personnel on UNC4736 social engineering TTPs observed at crypto industry conferences, NIST SI-4 (System Monitoring) — implement behavioral monitoring on IDE extension host processes as a standing detection capability, NIST AC-6 (Least Privilege) — enforce isolation of Solana private key material from developer workstation environments via hardware security modules or dedicated signing machines, CIS 6.3 (Require MFA for Externally-Exposed Applications) — require hardware security keys (FIDO2/YubiKey) for all governance signers, CIS 6.5 (Require MFA for Administrative Access) — enforce hardware MFA for all accounts with access to key management systems or on-chain governance, CIS 7.1 (Establish and Maintain a Vulnerability Management Process) — incorporate IDE extension vetting into vulnerability management process, CIS 5.4 (Restrict Administrator Privileges to Dedicated Administrator Accounts) — enforce dedicated, isolated signing machines for governance key operations, separate from developer workstations

Compensating: For a 2-person team with limited budget: (1) Hardware security keys — YubiKey 5 series (~$50 each) supports FIDO2 and can be used with Solana governance tooling; require physical key presence for all multisig signing operations. (2) Key isolation — designate one air-gapped or network-isolated machine (a $200 Raspberry Pi 4 or a wiped laptop) as the exclusive signing host; never install IDE extensions or connect to external networks on this machine. (3) IDE behavioral monitoring — deploy Sysmon with a custom rule alerting on 'extensionHost' processes making outbound TCP connections: add a Sysmon NetworkConnect rule filtering on Image containing 'extensionHost' or 'Code - OSS'. (4) Contributor vetting — create a simple checklist requiring: video call identity verification, government ID cross-check, LinkedIn/GitHub history review, and a 30-day probationary period before any governance access — document and sign off in your issue tracker. (5) Deploy Sigma rule

'proc_creation_susp_vscode_child_process' (available in SigmaHQ repository) converted to Windows Event Log format for IDE process monitoring without a SIEM.

Evidence: For the lessons-learned record and future detection improvement: (1) Compile a timeline mapping UNC4736 conference contact events (from attendee records) to first suspicious extension installation dates — this establishes the social engineering lead time and informs future vetting policy. (2) Preserve all malicious extension packages with SHA-256 hashes for submission to CISA, FS-ISAC, and internal threat intelligence feeds. (3) Document the exact on-chain transaction IDs and timestamps of the 12-minute fund drain for regulatory reporting and potential law enforcement referral. (4) Export and archive all GitHub commit history, PR approvals, and access grants made by the compromised developer identity during the six-month window — this establishes the full scope of code and access exposure beyond the financial loss.

Detection Guidance

No confirmed public IOCs (IPs, domains, hashes) for this campaign have been released in available T3 sources at time of analysis. Detection must rely on behavioral indicators. (1) IDE process anomalies: monitor for VSCode or Cursor IDE extension host processes (extensionHost, code-server child processes) initiating outbound TCP connections to non-approved destinations, writing files to %APPDATA%\Roaming\Microsoft\Windows\Start Menu\Programs\Startup or launchd plist locations on macOS, or accessing OS credential stores (Windows Credential Manager, macOS Keychain). (2) Governance access anomalies: alert on any Security Council or multisig signer addition not preceded by a documented approval workflow; monitor on-chain events for governance role changes using your protocol's event logs. (3) Social engineering indicators: flag any external contact requesting repository access, TestFlight app installations, or contributor onboarding who was introduced through a conference or informal channel rather than a formal process, particularly if representing a quantitative trading or investment firm. (4) MITRE T1036 masquerading: verify the legal entity and personnel identity of any firm seeking elevated access through business registry checks and video-verified identity, not email correspondence alone. If CrowdStrike (the attributing vendor for UNC4736) or CISA releases a formal advisory with IOCs, treat those as primary and supersede behavioral-only guidance.

Indicators of Compromise

Type	Value	Context	Confidence
DOMAIN	`[not publicly disclosed in available sources]`	No confirmed IOCs have been released in T3 sources covering this campaign at time of analysis. IOC release is pending formal vendor or government advisory.	LOW

Framework Mappings

MITRE-ATTACK

- **T1560** — Archive Collected Data
- **T1070.004** — File Deletion
- **T1036** — Masquerading
- **T1552** — Unsecured Credentials
- **T1195.001** — Compromise Software Dependencies and Development Tools

- **T1588.001** — Malware
- **T1204.002** — Malicious File
- **T1547** — Boot or Logon Autostart Execution
- **T1591.004** — Identify Roles
- **T1566** — Phishing
- **T1078** — Valid Accounts
- **T1566.003** — Spearphishing via Service
- **T1562** — Impair Defenses
- **T1657** — Financial Theft

NIST-800-53R5

- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **AT-2** — Literacy Training and Awareness
- **CA-7** — Continuous Monitoring
- **SC-7** — Boundary Protection
- **SI-8** — Spam Protection
- **AC-2** — Account Management
- **AC-6** — Least Privilege
- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **AU-9** — Protection of Audit Information
- **CM-6** — Configuration Settings
- **SI-10** — Information Input Validation
- **SI-7** — Software, Firmware, and Information Integrity
- **AC-3** — Access Enforcement
- **SR-2** — Supply Chain Risk Management Plan

OWASP-TOP10-2021

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

CIS-V8

- **16.10** — Apply Secure Design Principles in Application Architectures
- **6.1** — Establish an Access Granting Process
- **6.2** — Establish an Access Revoking Process
- **14.2** — Train Workforce Members to Recognize Social Engineering Attacks
- **15.1** — Establish and Maintain an Inventory of Service Providers

SOC2-TSC

- **CC6.1** — The entity implements logical access security software, infrastructure, and architectures over protected information assets
- **CC9.2** — Manages risks associated with vendors and business partners

HIPAA-SECURITY

- **164.312(a)(1)** — Access Control
- **164.308(a)(5)(i)** — Security Awareness and Training

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities
- **A.5.34** — Privacy and protection of personal information
- **A.5.21** — Managing information security in the ICT supply chain

NIST-CSF-2

- **GV.SC-01** — Cybersecurity supply chain risk management program

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1560	Archive Collected Data	Collection
T1070.004	File Deletion	Defense-Evasion
T1036	Masquerading	Defense-Evasion
T1552	Unsecured Credentials	Credential-Access
T1195.001	Compromise Software Dependencies and Development Tools	Initial-Access
T1588.001	Malware	Resource-Development
T1204.002	Malicious File	Execution
T1547	Boot or Logon Autostart Execution	Persistence
T1591.004	Identify Roles	Reconnaissance
T1566	Phishing	Initial-Access
T1078	Valid Accounts	Defense-Evasion
T1566.003	Spearphishing via Service	Initial-Access
T1562	Impair Defenses	Defense-Evasion
T1657	Financial Theft	Impact

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/drift-280m-crypto-th...	T3
North Korean Hackers Spent Six Months Infiltrating Drift Before $285 ...	https://finance.yahoo.com/markets/crypto/articles/north-korean-hack...	T3
Drift Protocol's $285m hack exposes social engineering threat to ...	https://crypto.news/drift-protocols-285m-hack-exposes-social-engine...	T3
Drift Protocol's $285M Solana Exploit Sparks DeFi Security Concerns	https://www.coca.xyz/post/drift-protocol-s-285m-solana-exploit-spar...	T3
Drift Protocol Rocked by $270 Million Exploit: Solana DeFi Under ...	https://blockchair.com/news/drift-protocol-rocked-by-270-million-ex...	T3

Generated 2026-04-06 18:18 UTC by TJS Security Command Center