INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-04-26 06:11 UTC

Pre-Stuxnet 'fast16' Lua-Based Cyber Sabotage Malware Uncovered, Targeting Engineering Calculation Software

THREAT CAMPAIGN | HIGH

SCC Item ID	SCC-CAM-2026-0223
Type	Threat Campaign
Severity	HIGH
Affected Products	High-precision calculation software (specific product unidentified; general engineering/scientific computation software, circa 2005)
Published	2026-04-25
Discovery Source	Gemini

Executive Summary

Researchers have identified a Lua-based malware framework called 'fast16,' believed to date to approximately 2005, designed to manipulate computational outputs in engineering calculation software rather than steal data. If verified, this discovery would place a sophisticated cyber sabotage capability targeting engineering systems at least five years before Stuxnet, reframing assumptions about when state-level or advanced actors began targeting operational technology. No corroboration from CISA, NVD, MITRE, or primary threat intelligence sources (Dragos, Mandiant, Claroty) has been identified as of publication. This finding is sourced from [original researcher/vendor name], and should be treated as unverified intelligence requiring confirmation from authoritative OT/ICS security bodies before operational response.

Technical Analysis

The 'fast16' framework is described as a Lua-scripted malware tool targeting high-precision engineering calculation software, with a sabotage objective: manipulating computational results rather than exfiltrating data. Alleged date of origin is circa 2005. No CVE has been assigned. Mapped weaknesses include CWE-345 (Insufficient Verification of Data Authenticity), CWE-284 (Improper Access Control), and CWE-506 (Embedded Malicious Code). MITRE ATT&CK technique mappings include T1195 (Supply Chain Compromise), T1059.007 (Command and Scripting Interpreter: JavaScript/Lua), T1036 (Masquerading), and T1565.001 (Stored Data Manipulation). No confirmed attribution, no CVE identifier, no CVSS score, and no patch or vendor advisory exists. Critically, the sources linked in the original discovery item do not reference 'fast16' and do not

substantiate any technical claims made about this malware. This item's source quality score is 0.516 and all linked sources are Tier 2 or Tier 3. The technical claims cannot be verified against primary authorities at this time.

Action Checklist

1. Step 0: Source Verification. Identify and review the original 'fast16' discovery report. If the researcher or threat intelligence firm cannot be named, or if the original source is unavailable, this item should be treated as preliminary reporting and assigned to threat intelligence monitoring only (no active operational investigation). Proceed to Step 1 only if the original source is credible and traceable.
2. Step 1: Contextualize, Do not initiate incident response based solely on this report. The discovery lacks corroboration from CISA, NVD, MITRE ATT&CK, or any primary ICS/OT security authority. Assign a threat intelligence analyst to track for corroborating reporting before escalating.
3. Step 2: Inventory, Identify any legacy engineering calculation software in your environment, particularly tools used in high-precision or safety-critical contexts (e.g., simulation, structural analysis, process control support). Flag systems running software from the 2003-2007 era that may not have received modern integrity controls.
4. Step 3: Detection (conditional), If your environment includes legacy engineering software with scripting engine support, review process execution logs for unexpected Lua interpreter activity (lua.exe, lua5x.exe, embedded Lua calls from calculation tools). Check file integrity monitoring logs for unexplained changes to calculation output files or configuration data.
5. Step 4: Validate Intelligence, Monitor CISA advisories (cisa.gov/ics-advisories), MITRE ATT&CK updates, and reputable ICS/OT threat intelligence sources (Dragos, Claroty, Mandiant) for corroborating reports on 'fast16.' Upgrade response posture only if primary sources confirm the finding.
6. Step 5: Post-Assessment, Use this report as a prompt to audit OT and engineering software environments for data integrity controls. Evaluate whether computational outputs from engineering tools are verified against known-good baselines. CWE-345 and CWE-506 exposures are broadly relevant to legacy OT software regardless of whether 'fast16' is confirmed.

IR / Forensic Enrichment

Triage Priority	DEFERRED
Escalation Criteria	Upgrade from deferred to urgent if CISA issues an ICS-CERT advisory referencing fast16 or the underlying Lua-based engineering sabotage research, if two independent ICS security vendors publish corroborating technical analysis with actionable IOCs, or if internal detection from Step 3 identifies actual Lua interpreter execution outside expected calculation tool process trees on any safety-critical or OT-adjacent engineering workstation.

Recovery Notes	Recovery actions are not yet applicable as no confirmed incident has been declared; however, if fast16 activity is subsequently confirmed in your environment, recovery must prioritize reestablishing trust in all computational outputs produced by affected engineering tools — this means rerunning calculations from verified clean inputs on isolated or reimaged systems and comparing results against known-good historical baselines before any outputs are used in safety-critical or operational decisions. Monitor calculation output files and Lua interpreter process execution for a minimum of 30 days post-remediation given fast16's reported design as a long-dwell sabotage tool rather than a rapid-exfiltration threat. Retain all forensic artifacts and output file hash logs for a minimum of one year to support potential future regulatory or legal review if the malware's historical activity is later attributed to infrastructure relevant to your organization.
Forensic Artifacts	Lua interpreter binary artifacts: File system search results for lua.exe, lua5x.exe, lua51.dll, lua5.1.dll in engineering workstation directories outside expected application install paths — fast16 as a Lua-based framework would require a Lua runtime present on the host, either standalone or embedded in the calculation tool; unexpected Lua binaries in temp directories, user profile directories, or calculation tool working folders are a primary indicator | Calculation output file hash deltas: SHA-256 hash manifests of engineering tool output files (.dat, .csv, .res, .out, .rpt) captured before and after calculation job execution — fast16's sabotage mechanism manipulates computational outputs rather than exfiltrating data, making unexplained hash changes to output files that do not correspond to logged job executions the most direct forensic artifact this malware class would produce | Process execution logs for anomalous parent-child relationships: Windows Security Event ID 4688 or Sysmon Event ID 1 records showing Lua interpreter processes (lua.exe, lua5x.exe) spawned by engineering calculation tool executables outside their documented scripting workflows, or spawned by unexpected parent processes (cmd.exe, powershell.exe, wscript.exe) on engineering workstations — this would indicate fast16-style script injection into the calculation tool's execution environment | Lua script files in unexpected locations: File system enumeration results for .lua and .luac (compiled Lua bytecode) files in engineering tool working directories, temp folders, or application data directories where Lua scripts are not expected per the tool's documented file structure — fast16 as a framework would persist malicious Lua scripts that hook into the calculation tool's scripting engine to intercept and modify outputs | Calculation tool configuration and plugin directory modifications: File modification timestamps and access logs (Windows Security Event ID 4663 if object access auditing is enabled, or AIDE change reports on Linux) for engineering tool plugin directories, configuration files, and any directory the tool scans for scripts or modules at startup — fast16's persistence mechanism would most likely involve placing a malicious Lua script in a location the legitimate calculation tool loads automatically, making modification timestamps in these directories that predate the investigation a key forensic anchor

Per-Action IR Details

Step 1: Contextualize — Do not initiate incident response based solely on this report. The discovery lacks corroboration from CISA, NVD, MITRE ATT&CK, or any primary ICS/OT security authority. Assign a threat intelligence analyst to track for corroborating reporting before escalating.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection & Analysis: Analysts must assess whether reported activity constitutes a credible incident before committing response resources; unverified third-party research requires corroboration before escalation.

Controls: NIST IR-4 (Incident Handling) — Requires organization to implement a structured handling capability that includes triage and prioritization before resource commitment, NIST IR-6 (Incident Reporting) — Establishes the

threshold criteria for what constitutes a reportable incident; unverified research does not yet meet this threshold, NIST SI-5 (Security Alerts, Advisories, and Directives) — Mandates monitoring of external security advisories and applying organizational judgment on credibility before acting, CIS 7.1 (Establish and Maintain a Vulnerability Management Process) — Requires a documented process for evaluating incoming threat intelligence before triggering remediation workflows

Compensating: Assign one analyst to create a tracking ticket in any free ticketing system (e.g., GitHub Issues, JIRA free tier, or even a shared spreadsheet) with the following fields: source publication, corroborating sources checked (CISA ICS-CERT feed via RSS at https://www.cisa.gov/uscert/ics/advisories, MITRE ATT&CK changelog), date of last check, and escalation threshold. Set a calendar reminder for weekly review. No SIEM required — this is a watch-and-track posture, not active detection.

Evidence: Before closing the tracking ticket or upgrading posture, document: (1) the original research publication metadata (author, outlet, date, SHA-256 hash of the PDF/article if downloadable); (2) a screenshot or export of MITRE ATT&CK technique coverage relevant to Lua-based in-process manipulation (closest current techniques: T1059.007 — Command and Scripting Interpreter: JavaScript/JScript as analog for embedded scripting, and T1565.001 — Stored Data Manipulation for output file tampering); (3) CISA ICS-CERT advisory archive search results showing absence of fast16 advisories as of the check date — this creates a dated audit trail of your due-diligence posture.

Step 2: Inventory — Identify any legacy engineering calculation software in your environment, particularly tools used in high-precision or safety-critical contexts (e.g., simulation, structural analysis, process control support). Flag systems running software from the 2003-2007 era that may not have received modern integrity controls.

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: Maintaining an accurate asset inventory of systems that could be targeted is a foundational preparation activity; for OT-adjacent engineering software, this includes understanding which legacy tools process safety-critical computational outputs.

Controls: NIST IR-4 (Incident Handling) — Preparation sub-phase requires knowing which assets are in scope before a threat can be assessed against them, NIST SI-7 (Software, Firmware, and Information Integrity) — Requires integrity verification tooling be applied to software; legacy engineering tools from 2003-2007 era are unlikely to have signed update chains or embedded integrity verification, NIST CM-8 (System Component Inventory) — Requires maintaining an accurate inventory of system components; engineering calculation software in OT-adjacent environments is frequently undocumented, CIS 1.1 (Establish and Maintain Detailed Enterprise Asset Inventory) — Mandates hardware asset inventory; extend this to software assets per CIS 2.1, CIS 2.1 (Establish and Maintain a Software Inventory) — Specifically requires tracking all licensed software; legacy engineering tools from the 2003-2007 era that embed Lua scripting engines are the direct target population for fast16, CIS 2.2 (Ensure Authorized Software is Currently Supported) — Engineering calculation software from 2003-2007 that is no longer vendor-supported represents elevated risk in this context

Compensating: Run the following on Windows hosts to enumerate installed software with install dates: `Get-WmiObject -Class Win32_Product | Select-Object Name, Version, InstallDate | Where-Object {$_.InstallDate -lt '20080101'} | Export-Csv legacy_software.csv -NoTypeInformation`. On Linux/Unix engineering workstations: `dpkg-query -W -f='${Package} ${Version} ${Status} ' > installed_packages.txt` combined with `rpm -qa --queryformat '%{NAME} %{VERSION} %{INSTALLTIME:date} '` for RPM-based systems. Cross-reference results against a manually maintained spreadsheet of known engineering calculation tools (MATLAB, Mathematica, legacy FEA tools, custom in-house calculation frameworks). Flag any that shipped with embedded Lua 5.0 or 5.1 interpreters.

Evidence: Before actioning remediation, capture: (1) full software inventory exports with install dates and version strings from all engineering workstations and OT-adjacent systems — these serve as the baseline for any subsequent integrity comparison; (2) file system listings of directories containing calculation output files (e.g., .csv, .dat, .out, .res, .rpt file types in engineering tool working directories) with timestamps and file hashes (use `certutil -hashfile SHA256` on Windows or `sha256sum` on Linux) — fast16's sabotage mechanism targets computational outputs, so pre-investigation baseline hashes are critical forensic anchors; (3) a list of any embedded Lua DLLs or standalone Lua interpreter binaries found on engineering hosts (search for lua5.1.dll, lua51.dll, lua.exe, lua5x.exe).

Step 3: Detection (conditional) — If your environment includes legacy engineering software with scripting engine support, review process execution logs for unexpected Lua interpreter activity (lua.exe, lua5x.exe, embedded Lua calls from calculation tools). Check file integrity monitoring logs for unexplained changes to calculation output files or configuration data.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection & Analysis: Analysts should correlate process execution telemetry with file modification events to identify whether Lua interpreter invocations outside normal calculation software workflows represent anomalous scripting activity consistent with the fast16 sabotage-by-script-manipulation pattern.

Controls: NIST SI-4 (System Monitoring) — Requires monitoring for unauthorized activity; embedded Lua execution outside the parent calculation application's expected process tree is the specific anomaly to detect here, NIST AU-2 (Event Logging) — Requires logging of events relevant to security; process creation events for Lua interpreter binaries and file modification events on calculation output directories are the relevant event types, NIST AU-6 (Audit Record Review, Analysis, and Reporting) — Requires periodic review of audit logs for indicators; review cadence should be daily for any engineering systems flagged in Step 2, CIS 8.2 (Collect Audit Logs) — Requires audit logging to be enabled; process execution logging is not enabled by default on Windows and must be configured via Sysmon or audit policy

Compensating: Deploy Sysmon (free, Microsoft Sysinternals) with a configuration that captures Event ID 1 (Process Create) filtered for: Image contains 'lua', ParentImage matches calculation tool executables identified in Step 2 inventory. Use the SwiftOnSecurity Sysmon config as a baseline (github.com/SwiftOnSecurity/sysmon-config) and add a custom rule targeting Lua process names. For file integrity monitoring without a SIEM, use AIDE (Linux) or Windows built-in `auditpol /set /subcategory:'File System' /success:enable /failure:enable` combined with a PowerShell scheduled task that hashes all files in calculation output directories nightly and diffs against the prior day's baseline: `Get-FileHash -Path 'C:\EngineeringOutputs\*' -Algorithm SHA256 | Export-Csv daily_hash_$(Get-Date -Format yyyyMMdd).csv`. Alert on any hash change to output files that did not correspond to a logged calculation job execution.

Evidence: Capture before beginning log review: (1) Windows Security Event Log Event ID 4688 (Process Creation) — filter for processes where 'NewProcessName' contains 'lua' or where the parent process is a known calculation tool and the child process is unexpected; (2) Sysmon Event ID 1 logs showing full command-line arguments for any Lua interpreter invocations — fast16's mechanism would likely pass a script path as an argument, revealing the malicious script location; (3) file modification timestamps and hash deltas on calculation output files (.dat, .csv, .res, .out) from engineering tool working directories — the sabotage mechanism manipulates outputs rather than stealing data, so a file that was modified without a corresponding logged calculation run is the primary forensic indicator; (4) Windows Security Event ID 4663 (Object Access — File) on calculation output directories if object access auditing is enabled.

Step 4: Validate Intelligence — Monitor CISA advisories (cisa.gov/ics-advisories), MITRE ATT&CK updates, and reputable ICS/OT threat intelligence sources (Dragos, Claroty, Mandiant) for corroborating reports on 'fast16.' Upgrade response posture only if primary sources confirm the finding.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection & Analysis: Integrating external cyber threat intelligence (CTI) into incident analysis — specifically verifying that a reported threat meets the organization's incident declaration threshold before escalating — is a core detection-phase activity aligned with DE.AE-07 and DE.AE-08 from the NIST CSF 2.0 mapping in 800-61r3.

Controls: NIST SI-5 (Security Alerts, Advisories, and Directives) — Requires receiving and acting on security advisories from external organizations; CISA ICS-CERT is the primary authority for OT/ICS threats including any future fast16 confirmation, NIST IR-4 (Incident Handling) — Requires applying incident criteria to analyzed activity before declaring an incident; fast16 does not yet meet incident declaration criteria absent primary source corroboration, NIST RA-3 (Risk Assessment) — Requires assessing the likelihood and impact of threats; intelligence validation is the mechanism for refining likelihood estimates for unconfirmed research findings, CIS 7.1 (Establish and Maintain a Vulnerability Management Process) — Requires a documented process for tracking and acting on threat intelligence; the watch-and-track workflow initiated in Step 1 is the operationalization of this safeguard for fast16

Compensating: Subscribe to CISA ICS-CERT advisories via free RSS feed (no account required). Set a Google Alert or RSS monitor for the search term 'fast16 malware' and 'Lua engineering sabotage' using a free RSS aggregator (e.g.,

Feedly free tier). Create a one-page Intelligence Requirements (IR) document listing the specific corroboration criteria that would trigger posture upgrade: (a) CISA ICS-CERT advisory referencing fast16 or the underlying research, (b) MITRE ATT&CK technique addition referencing Lua-based engineering software manipulation, or (c) two independent ICS security vendors (e.g., Dragos + Claroty, or Claroty + Mandiant) publishing corroborating technical analysis with IOCs. Review this document at the weekly cadence established in Step 1.

Evidence: Document and preserve: (1) dated exports or screenshots of CISA ICS-CERT advisory search results for 'fast16,' 'Lua malware,' and 'engineering calculation software' at each weekly check — these create an audit trail showing your organization actively monitored for corroboration; (2) any MITRE ATT&CK technique changelog entries (available at attack.mitre.org/resources/changelog) showing additions related to embedded scripting engine abuse in engineering or OT software contexts; (3) if Dragos, Claroty, or Mandiant publish corroborating reports, archive the full report with its publication hash — these become primary source evidence that would trigger the posture upgrade decision and should be retained per NIST AU-11 (Audit Record Retention) requirements for your organization's defined period.

Step 5: Post-Assessment — Use this report as a prompt to audit OT and engineering software environments for data integrity controls. Evaluate whether computational outputs from engineering tools are verified against known-good baselines. CWE-345 and CWE-506 exposures are broadly relevant to legacy OT software regardless of whether 'fast16' is confirmed.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: Even when a reported threat does not result in a declared incident, post-assessment activities including lessons learned and control gap analysis are appropriate; fast16's sabotage-by-output-manipulation mechanism exposes a class of integrity control gaps in engineering software environments that should be addressed independent of whether this specific malware is confirmed.

Controls: NIST SI-7 (Software, Firmware, and Information Integrity) — Directly addresses the CWE-345 (Insufficient Verification of Data Authenticity) exposure: requires integrity verification tools to detect unauthorized changes to software and data; legacy engineering calculation tools from the 2003-2007 era typically lack signed output verification, making this the highest-priority control gap surfaced by fast16, NIST SI-2 (Flaw Remediation) — CWE-506 (Embedded Malicious Code) is a flaw class requiring organizational remediation processes; for legacy tools without vendor support, the remediation process defaults to compensating controls and isolation, NIST IR-8 (Incident Response Plan) — Post-assessment findings should feed back into the IR plan; specifically, the absence of integrity controls on engineering calculation outputs should be documented as a gap requiring plan updates, NIST AU-11 (Audit Record Retention) — Calculation output files and their hash baselines should be retained per policy to support future forensic comparison if fast16 or a similar sabotage campaign is later confirmed, CIS 7.2 (Establish and Maintain a Remediation Process) — Requires a risk-based remediation strategy; CWE-345 and CWE-506 exposures in unsupported legacy engineering software should be formally risk-rated and assigned remediation timelines in this process, CIS 3.2 (Establish and Maintain a Data Inventory) — Computational output files from engineering and safety-critical tools are sensitive data that should be inventoried and protected; fast16's targeting of output files rather than source data makes this control directly relevant

Compensating: For a 2-person team with no budget: implement a hash-verification wrapper script for engineering calculation jobs. On Windows: a PowerShell wrapper that (1) records input file hashes before job execution, (2) runs the calculation tool, (3) records output file hashes after execution, and (4) writes the delta to a tamper-evident log (append-only file with NTFS permissions restricting deletion). On Linux: a bash wrapper using `sha256sum` before and after execution, writing to a log owned by root with permissions 644 (writable only by root). Store daily output hash manifests in a separate location from the calculation tool's working directory — fast16-style malware that manipulates outputs would need to also tamper with the verification log to evade this control. Use AIDE (Advanced Intrusion Detection Environment, free) on Linux engineering hosts to monitor calculation tool binaries and configuration files for unauthorized modification, directly addressing the CWE-506 embedded malicious code risk.

Evidence: Artifacts to collect and retain as part of the post-assessment: (1) the output of the software inventory from Step 2, annotated with whether each tool has vendor-supported integrity verification for its output files — this becomes the gap analysis baseline; (2) results of a manual review of calculation tool configuration directories for any unexpected .lua, .luac, or script files that should not be present per the application's documented file structure; (3) a risk register entry documenting the CWE-345 and CWE-506 exposure classes, the affected engineering tool population identified in

Step 2, and the compensating controls implemented — this satisfies NIST RA-3 (Risk Assessment) documentation requirements and provides evidence for future audit inquiries about how the organization responded to the fast16 research report.

Detection Guidance

No verified IOCs, file hashes, network indicators, or behavioral signatures for 'fast16' are available from authoritative sources. Detection guidance below is based solely on mapped ATT&CK techniques and should be treated as general hygiene, not confirmed threat-specific detection. In environments where engineering software legitimately uses embedded Lua for scripting and data processing, these monitoring recommendations may generate false positives. Baseline your environment first, then implement alerts on deviations from known-good behavior. For T1059.007 (Lua scripting interpreter abuse): monitor for Lua interpreter execution spawned from engineering or calculation software processes; alert on lua.exe or embedded Lua calls that are not part of a known baseline. For T1565.001 (Stored Data Manipulation): implement file integrity monitoring on calculation output files, configuration files, and result logs in engineering software directories; alert on unexpected modifications outside of normal workflow windows. For T1036 (Masquerading): review process names and loaded modules in engineering workstation environments for binaries mimicking legitimate calculation software components. For T1195 (Supply Chain Compromise): if the original software distribution is suspected, verify installer hashes against vendor-published checksums. No confirmed IOC patterns exist for this item. Do not treat detection hits as confirmed 'fast16' activity without further investigation.

Framework Mappings

MITRE-ATTACK

- **T1195** — Supply Chain Compromise
- **T1059.007** — JavaScript
- **T1036** — Masquerading
- **T1565.001** — Stored Data Manipulation

NIST-800-53R5

- **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
- **AC-3** — Access Enforcement

OWASP-TOP10-2021

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

CIS-V8

- **2.5** — Allowlist Authorized Software
- **6.1** — Establish an Access Granting Process

- **6.2** — Establish an Access Revoking Process

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

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1195	Supply Chain Compromise	Initial-Access
T1059.007	JavaScript	Execution
T1036	Masquerading	Defense-Evasion
T1565.001	Stored Data Manipulation	Impact

Sources

Source	URL	Tier
Human-understandable explanation for software vulnerability ...	https://www.sciencedirect.com/science/article/pii/S0164121225001232	T3
CWE - Common Weakness Scoring System (CWSS)	https://cwe.mitre.org/cwss/	T3
[PDF] Measuring the accuracy of software vulnerability assessments	https://air.unimi.it/retrieve/dfa8b9a1-440b-748b-e053-3a05fe0a3a96/...	T3
Stop Using Vulnerability Counts to Measure Software Security	https://cacm.acm.org/opinion/stop-using-vulnerability-counts-to-mea...	T3
A Survey of Vulnerability Detection Techniques and Insights - arXiv	https://arxiv.org/html/2502.07049v1	T2

Generated 2026-04-26 06:11 UTC by TJS Security Command Center