INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-05-27 14:07 UTC

EvidenceForge Closes the Synthetic Log Gap: Cisco Talos Ships Causal, Cross-Format Training Data Generator

SECURITY ANALYSIS | **LOW** | CVSS 2.5

SCC Item ID	SCC-STY-2026-0158
Type	Security Analysis
Severity	LOW
CVSS Base Score	2.5
Affected Products	EvidenceForge (Cisco Talos open-source tool); supports Windows Security Events, Sysmon, EDR/XDR telemetry, Linux syslog, Zeek, Snort, firewall and proxy logs (20+ formats)
Published	2026-05-27T10:00:47+00:00
Discovery Source	Rss:T1 Threatintel

Executive Summary

Cisco Talos released EvidenceForge, an open-source tool that generates causally consistent synthetic security logs across more than 20 log formats simultaneously, solving a longstanding data quality problem that has constrained ML-based detection development. Security teams building or validating detection models have historically relied on either scarce production data or synthetic logs that fail cross-format correlation checks, limiting model accuracy. EvidenceForge signals a maturing investment in detection engineering infrastructure, giving both enterprise teams and the broader research community a credible path to labeled training data without exposing sensitive production telemetry.

Technical Analysis

Detection engineering and ML-based security tooling share a foundational problem: labeled, high-fidelity training data is rare, and the alternatives carry real costs. Production logs contain sensitive data, making broad sharing legally and operationally complicated. Existing synthetic generators solve the volume problem but not the consistency problem, they emit events per format without maintaining shared state, so a Windows Security Event referencing a process has no coherent relationship to the Sysmon event, EDR telemetry, or network flow that should accompany it. A trained analyst, or a well-tuned detection model, spots that incoherence immediately.

EvidenceForge addresses this by building from a single canonical event model. When the tool generates a process execution event, it propagates the associated artifacts, process tree relationships, parent-child PIDs,

network connections, authentication events, file writes, across every relevant log format simultaneously. The result is synthetic telemetry where a Sysmon ProcessCreate event, its corresponding Windows Security Event 4688, the Zeek conn.log entry for the spawned network connection, and the EDR telemetry record all reference consistent identifiers and timestamps. That cross-format coherence is what prior tools could not reliably produce.

The MITRE ATT&CK techniques listed in the item data (T1078 Valid Accounts, T1021 Remote Services, T1059 Command and Scripting Interpreter, T1543 Create or Modify System Process, T1558 Steal or Forge Kerberos Tickets, T1046 Network Service Discovery, T1190 Exploit Public-Facing Application) represent the technique coverage the tool can model, common adversary behaviors that detection teams most need labeled examples of. This is not incidental: the hardest detection engineering problems cluster around exactly these techniques because they involve legitimate tool abuse, living-off-the-land patterns, and behaviors that blend into normal administrative activity.

Three primary use cases emerge from the tool's design. First, labeled dataset generation for ML and detection model training, where teams need thousands of coherent positive and negative examples across technique classes. Second, detection rule validation in non-production environments, where a new Sigma rule or SIEM query can be tested against synthetic telemetry that resembles real attacker behavior without requiring a live threat or a production log export. Third, threat hunter training, where analysts can work through realistic multi-source scenarios without access to sensitive organizational data or a live range environment.

The open-source release via the Cisco-Talos GitHub organization means the tool is immediately available for community inspection, contribution, and integration into existing detection pipelines. That transparency also means the adversary community can study it, which creates a secondary consideration: synthetic data quality improvements raise the bar for red teams and adversary emulation exercises to match the fidelity that defenders now expect in training environments. The tool does not introduce a vulnerability. Its strategic value is in closing the data infrastructure gap that has consistently limited detection engineering maturity.

Action Checklist

1. Step 1: Assess relevance, determine whether your security engineering, detection, or ML teams are currently constrained by training data availability or synthetic data quality limitations
2. Step 2: Evaluate detection coverage gaps, cross-reference your current detection rule coverage against the MITRE ATT&CK techniques EvidenceForge can model (T1078, T1021, T1059, T1543, T1558, T1046, T1190) and identify where labeled test data is absent
3. Step 3: Review synthetic data practices, audit whether existing detection rule validation and model training workflows rely on synthetic logs, and assess whether cross-format consistency is verified as part of that process (NIST SI-4 supports continuous monitoring and detection validation requirements)
4. Step 4: Pilot in detection engineering workflows, assign a detection engineer to evaluate EvidenceForge for one specific use case: rule validation, model training, or analyst training exercises; document findings against your current toolchain
5. Step 5: Monitor community development, track the Cisco-Talos/EvidenceForge GitHub repository for new format support, community contributions, and integration patterns from peer organizations; the open-source model means capability will expand through community iteration

IR / Forensic Enrichment

Triage Priority	DEFERRED
Escalation Criteria	Escalate from deferred to standard priority if your organization's ML-based detection models are actively producing false negatives on any of the seven EvidenceForge-modeled techniques (T1078, T1021, T1059, T1543, T1558, T1046, T1190) and the root cause is confirmed to be insufficient or cross-format-inconsistent training data, as this represents a measurable detection capability gap with direct operational impact.
Recovery Notes	EvidenceForge is a development and training tool, not a production detection component, so standard incident recovery steps do not apply. Post-pilot, verify that any synthetic datasets generated are stored in a controlled, access-restricted directory and not inadvertently ingested into production SIEM pipelines as real telemetry — synthetic log injection into live monitoring systems would corrupt alert fidelity. Monitor the detection rule performance metrics (true-positive rate, false-negative rate) for the specific ATT&CK techniques addressed during the pilot for 30 days post-integration to confirm measurable coverage improvement.
Forensic Artifacts	EvidenceForge GitHub repository commit history and release tags — preserving the specific version (`git log --oneline`) used during any detection rule validation or model training exercise creates an auditable record tying synthetic data quality to a known tool state | Synthetic dataset file hashes (SHA-256) and generation configuration files — hashing all EvidenceForge output files before use in model training or rule validation establishes tamper-evident provenance, critical if synthetic data is later cited in compliance audits or red team exercise reports | Cross-format consistency validation logs — records of whether synthetic Windows Security Event ID 4624/4688 entries, co-generated Sysmon Event ID 1/3 entries, and corresponding Zeek conn.log/dns.log entries share coherent timestamps, source/destination IPs, and process lineage for the same simulated session (absence of this artifact is evidence of the pre-EvidenceForge data quality gap) | Detection rule test results against synthetic data — sigma-cli or equivalent output logs showing which Sigma rules fired (true positives), which failed to fire (false negatives), and at what rate, scoped to the seven ATT&CK techniques EvidenceForge models (T1078, T1021, T1059, T1543, T1558, T1046, T1190) | ATT&CK Navigator coverage layer export — a dated JSON export of your ATT&CK Navigator layer showing technique coverage before and after EvidenceForge pilot integration, providing a before/after artifact that documents measurable detection coverage improvement attributable to improved synthetic training data quality

Per-Action IR Details

Step 1: Assess relevance — determine whether your security engineering, detection, or ML teams are currently constrained by training data availability or synthetic data quality limitations

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: Establishing IR capability, tools, and training data infrastructure

Controls: NIST SI-4 (System Monitoring) — ensures detection systems are validated with sufficient, representative training data, NIST CA-2 (Control Assessments) — supports periodic assessment of detection capability gaps, CIS 7.1 (Establish and Maintain a Vulnerability Management Process) — includes assessing detection coverage adequacy as part of the broader vulnerability posture, CIS 8.2 (Collect Audit Logs) — requires audit log coverage that presupposes detection models are trained against realistic, cross-format telemetry

Compensating: For a 2-person team without a formal ML pipeline: run a manual coverage gap audit by listing your active Sigma rules (https://github.com/SigmaHQ/sigma) and mapping each to a MITRE ATT&CK technique. Use the ATT&CK Navigator (free, browser-based) to visualize which techniques — specifically T1078, T1021, T1059, T1543, T1558, T1046, T1190 — have zero rule coverage. Document the count of uncovered techniques as your baseline constraint metric before evaluating EvidenceForge.

Evidence: This step is a planning activity, not a reactive forensic action. No pre-step evidence capture is required. Document the current state of your detection rule inventory (e.g., export active Sigma rule list to CSV with technique mappings) as a baseline artifact before any EvidenceForge evaluation begins, so post-pilot improvement is measurable.

Step 2: Evaluate detection coverage gaps — cross-reference your current detection rule coverage against the MITRE ATT&CK techniques EvidenceForge can model (T1078, T1021, T1059, T1543, T1558, T1046, T1190) and identify where labeled test data is absent

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: Detection capability assessment and tooling readiness

Controls: NIST SI-4 (System Monitoring) — validates that monitoring coverage spans the techniques EvidenceForge targets, NIST RA-3 (Risk Assessment) — supports structured identification of detection gaps as organizational risk, NIST CA-7 (Continuous Monitoring) — requires ongoing validation that detection rules remain effective against current technique coverage, CIS 7.1 (Establish and Maintain a Vulnerability Management Process) — detection gap analysis is an upstream dependency of effective vulnerability prioritization, CIS 8.2 (Collect Audit Logs) — log collection must cover the sources EvidenceForge emits (Windows Security Events, Sysmon, Zeek, Snort, proxy) for rule validation to be meaningful

Compensating: Use ATT&CK Navigator (free, offline-capable at https://mitre-attack.github.io/attack-navigator/) to load your Sigma rule set and color-code the seven EvidenceForge-modeled techniques: T1078 (Valid Accounts), T1021 (Remote Services), T1059 (Command and Scripting Interpreter), T1543 (Create or Modify System Process), T1558 (Steal or Forge Kerberos Tickets), T1046 (Network Service Discovery), T1190 (Exploit Public-Facing Application). Any technique with no colored cell and no labeled training data in your dataset is a confirmed gap requiring EvidenceForge-generated synthetic logs.

Evidence: Before running the gap analysis, export your current active detection rule list with last-validated dates. For Sigma-based environments, run: `python sigmac --list-backends` to confirm which log formats your rules currently target, and compare against EvidenceForge's 20+ supported formats (Windows Security Events, Sysmon, EDR/XDR telemetry, Linux syslog, Zeek, Snort, firewall, proxy). Document any format in EvidenceForge's output scope that has zero active Sigma rules as an artifact of this assessment.

Step 3: Review synthetic data practices — audit whether existing detection rule validation and model training workflows rely on synthetic logs, and assess whether cross-format consistency is verified as part of that process (NIST SI-4 supports continuous monitoring and detection validation requirements)

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: Validating detection tooling and ensuring training data integrity

Controls: NIST SI-4 (System Monitoring) — requires that monitoring systems are validated to detect the events they are configured to detect; synthetic data quality directly affects this assurance, NIST CA-2 (Control Assessments) — auditing synthetic data practices is an assessment activity that validates whether detection controls function as intended, NIST SA-11 (Developer Testing and Evaluation) — applies to detection model development workflows using synthetic data as test inputs, CIS 7.2 (Establish and Maintain a Remediation Process) — detection model remediation requires high-quality labeled data; this audit surfaces whether that dependency is currently met

Compensating: For teams generating synthetic logs manually or via scripts: run a cross-format consistency check by replaying your existing synthetic Windows Security Event logs alongside Zeek conn.log entries for the same simulated session and verify that source IP, destination IP, timestamp delta, and protocol match across both formats. A mismatch (e.g., a synthetic T1021 lateral movement event appearing in Windows Event ID 4624 logs but absent from the corresponding Zeek conn.log) is the exact cross-format incoherence EvidenceForge is designed to eliminate. Document each mismatch found as a workflow gap artifact.

Evidence: Before the audit, preserve a snapshot of your current synthetic dataset: record file hashes (sha256sum on Linux, `Get-FileHash` in PowerShell on Windows) of existing synthetic log files, their generation scripts, and any ML model training manifests. This establishes a pre-EvidenceForge baseline. If synthetic logs were generated by a prior tool or manual process, archive one representative sample per format (e.g., one synthetic Windows Security Event XML, one Zeek conn.log snippet, one Sysmon Event ID 1 entry) to enable before/after causal consistency comparison

after EvidenceForge is piloted.

Step 4: Pilot in detection engineering workflows — assign a detection engineer to evaluate EvidenceForge for one specific use case: rule validation, model training, or analyst training exercises; document findings against your current toolchain

NIST Phase: Preparation

Reference: NIST 800-61r3 §2 — Preparation: Tool evaluation, capability building, and documentation of detection infrastructure

Controls: NIST SI-4 (System Monitoring) — pilot directly supports improved monitoring fidelity by validating that detection rules perform correctly against causally consistent multi-format synthetic telemetry, NIST SA-11 (Developer Testing and Evaluation) — structured pilot of EvidenceForge against a defined use case is a testing and evaluation activity within the detection engineering development lifecycle, NIST CM-4 (Impact Analysis) — piloting in an isolated workflow (not production) aligns with impact analysis requirements before broader toolchain integration, CIS 7.4 (Perform Automated Application Patch Management) — EvidenceForge is an open-source dependency; the pilot should include establishing a process for tracking releases via the Cisco-Talos GitHub repo

Compensating: For a 2-person team: scope the pilot to Sysmon-based rule validation only. Clone the EvidenceForge repository, generate a synthetic Sysmon dataset modeling T1059.001 (PowerShell execution), and run your existing Sysmon-targeted Sigma rules against the output using sigma-cli (`sigma convert -t splunk -r rules/synthetic_sysmon.json`). If you lack a SIEM, pipe the output through `grep` or `jq` against the synthetic JSON logs. Document: (1) whether Sigma rules fired as expected, (2) whether the synthetic Sysmon Event ID 1 entries were causally linked to network events in the co-generated Zeek log, and (3) any false negative rate observed.

Evidence: Before the pilot, document your current detection rule true-positive and false-positive rates for the chosen use case (e.g., T1059 PowerShell detections in your Sysmon pipeline) using production data as the baseline. Preserve the EvidenceForge version hash (`git log --oneline -1` in the cloned repo) and the configuration parameters used to generate synthetic data. This creates a reproducible audit trail tying pilot results to a specific EvidenceForge commit, which is critical if the tool's output is later used to certify rule coverage for compliance or audit purposes.

Step 5: Monitor community development — track the Cisco-Talos/EvidenceForge GitHub repository for new format support, community contributions, and integration patterns from peer organizations; the open-source model means capability will expand through community iteration

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity: Lessons learned, capability improvement, and threat intelligence sharing

Controls: NIST SI-5 (Security Alerts, Advisories, and Directives) — monitoring the EvidenceForge repository for new releases and community advisories is analogous to tracking security alerts for detection tooling dependencies, NIST PM-16 (Threat Awareness Program) — community tracking of EvidenceForge development patterns from peer organizations constitutes threat intelligence sharing that strengthens the broader detection community, NIST SA-22 (Unsupported System Components) — tracking EvidenceForge's release cadence ensures the organization does not rely on an unmaintained version for detection validation, CIS 7.1 (Establish and Maintain a Vulnerability Management Process) — open-source tool dependencies must be tracked for updates and community-reported issues as part of ongoing vulnerability management

Compensating: Set up a no-cost GitHub watch on Cisco-Talos/EvidenceForge (Releases + Issues notifications) using a team-shared GitHub account. Additionally, configure a free RSS feed from the repo's releases page (`https://github.com/Cisco-Talos/EvidenceForge/releases.atom`) into your team communication channel or email. For peer organization integration patterns, monitor the Sigma community Slack and the MITRE ATT&CK Evaluations program announcements, where detection engineering teams commonly share toolchain integrations. Review new Issues and PRs monthly for community-reported cross-format consistency bugs that could affect synthetic data reliability.

Evidence: Establish a change log artifact: after each quarterly review of the EvidenceForge repository, document the current release version, any new log formats added (comparing against the prior quarter's supported format list), and any community-reported issues tagged as 'data-quality' or 'cross-format' in the GitHub Issues tracker. This provides an

audit trail demonstrating active governance of an open-source detection toolchain dependency, which supports NIST SA-22 compliance and justifies continued or discontinued use in your detection engineering workflow.

Detection Guidance

EvidenceForge is a defensive tooling release, not a threat event, so there are no adversary IOCs to hunt. The detection engineering value is prospective: teams should assess how the tool can improve the quality of existing detection content.

For organizations building or maintaining ML-based detection models, the relevant audit question is whether current training datasets include coherent cross-format examples for the technique classes EvidenceForge covers. Gaps in T1059 (scripting interpreter abuse) and T1078 (valid account misuse) training data are particularly common and particularly consequential, as these techniques underpin the majority of post-compromise lateral movement activity.

For detection rule validation, teams should verify that Sigma rules and SIEM queries targeting T1021 (remote services), T1543 (service creation and modification), and T1558 (Kerberos ticket abuse) are tested against multi-source telemetry, not single-log queries, before promotion to production. EvidenceForge's cross-format consistency makes it a candidate for this validation step.

For threat hunter training programs, the tool enables scenario construction that does not require production data access. Teams can assess whether current training exercises use coherent multi-source scenarios or rely on isolated log samples, which limits an analyst's ability to practice cross-format correlation, the core skill in identifying legitimate-tool abuse patterns.

Audit logging controls relevant to the technique coverage: NIST AU-2 (event logging), AU-3 (content of audit records), and AU-12 (audit record generation) define the logging baseline that EvidenceForge-generated data should mirror. If synthetic data does not reflect what your actual AU-2 logging configuration produces, training and validation results will not transfer accurately to production. CIS 8.2 (Collect Audit Logs) provides the IG1 baseline for ensuring logging is enabled across the formats EvidenceForge supports before synthetic data is used to validate detection coverage.

Framework Mappings

MITRE-ATTACK

- **T1078** — Valid Accounts
- **T1021** — Remote Services
- **T1059** — Command and Scripting Interpreter
- **T1543** — Create or Modify System Process
- **T1558** — Steal or Forge Kerberos Tickets
- **T1046** — Network Service Discovery
- **T1190** — Exploit Public-Facing Application

NIST-800-53R5

- **AC-2** — Account Management
- **AC-6** — Least Privilege

- **IA-2** — Identification and Authentication (Organizational Users)
- **IA-5** — Authenticator Management
- **AC-17** — Remote Access
- **AC-3** — Access Enforcement
- **CM-7** — Least Functionality
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **SI-7** — Software, Firmware, and Information Integrity
- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation

CIS-V8

- **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
T1078	Valid Accounts	Defense-Evasion
T1021	Remote Services	Lateral-Movement
T1059	Command and Scripting Interpreter	Execution
T1543	Create or Modify System Process	Persistence
T1558	Steal or Forge Kerberos Tickets	Credential-Access
T1046	Network Service Discovery	Discovery
T1190	Exploit Public-Facing Application	Initial-Access

Sources

Source	URL	Tier
Cisco Talos Blog	https://blog.talosintelligence.com/introducing-evidenceforge-synthe...	T3
EvidenceForge/docs/reference/EVIDENCE_FORMATS.md at main	https://github.com/Cisco-Talos/EvidenceForge/blob/main/docs/referen...	T3

Source	URL	Tier
Cisco-Talos/EvidenceForge - GitHub	https://github.com/Cisco-Talos/EvidenceForge	T3
Cisco Talos Threat Intelligence Services	https://www.cisco.com/site/us/en/products/security/talos/threat-int...	T3
Why Cisco Talos Matters. A Lot. - Network Solutions	https://www.nsi1.com/blog/why-cisco-talos-matters-a-lot	T3

Generated 2026-05-27 14:07 UTC by TJS Security Command Center