Inception Security companion appendix · SonicWall SSLVPN SAML bypass (CVE-2026-19868) · exact-v8 · source COMPANION.md sha256 652a4f4901132bb5cd33b6d69207813a6c1f647ecccc52e947475bb8265f64b9 · printable PDF for Wix Media hosting after Eric GO · not a substitute for the ZIP

SonicWall evidence contract and four-platform preparation

Start here

These are normalized hunting templates, not stock vendor connector searches. Use an audited adapter matching the explicit contract below. The optional `sonicwall-reference-v7.zip` includes reference_checks.py, illustrative JSON and self-contained tests. It needs only Python's standard library; the queries do not call it at runtime. Equivalent independently validated preparation is acceptable. Neither this program nor a zero-gap ledger establishes source truth, correct parsing or SIEM execution.

1. Preserve original bytes and **all transport receipts**, independently counted before classification. Inventory actual VPN, relevant IdP/application and endpoint populations, including silent sources, separately for each tenant. Audit exclusions, timestamp parsing, retention, historical policy/identity and session/device bindings.
2. Prepare one finite, closed-schema batch. Atomic structural admission rejects malformed identity/type, unknown kind, ambiguous transformation and collisions anywhere in the supplied population. Do not load any event or lookup from a rejected batch. Unknown decision evidence remains a hunt gap, never a clean negative.
3. With normal deployment authorization, load dedicated, unmixed event and ledger datasets. Load the two minimal Elastic lookup indexes separately. Follow the platform clock contract below. This packet performs **no network requests or resource creation**.
4. Obtain a genuine complete export independently, using selectors covering the entire actual native-time population. Compare exact logical row content/types/multiplicities, source set, all counts, lookup schemas/content and native timestamps. Do not generate a purported export from the prepared file. Counts alone are insufficient.
5. Validate target-version syntax and actual job completion, fan-out, memory/result limits and partitions. Only after those checks and independent source evidence are cleared can an analyst interpret a result or negative. `EXTERNAL_VALIDATION_REQUIRED` is not approval; `BLOCKED` forbids interpretation.

Optional example command: `python3 reference_checks.py examples/input-SYNTHETIC.json output-new`. Every invocation needs a new directory. Structural failure exits 2 and emits no event/lookup rows; do not consume partial files. Only successful complete local file generation writes COMPLETE.json, whose native_approval is false. It does not certify any source or engine. Errors and notes are local evidence; archive the original inputs independently.

Output destinations

Logical file	Splunk	Elastic	CrowdStrike CQL	Microsoft Sentinel KQL
events.json	index sonicwall_hunt	sonicwall-hunt	dataset sonicwall_hunt	SonicWallHunt_CL
coverage.json	index sonicwall_coverage	sonicwall-coverage	dataset sonicwall_coverage	SonicWallCoverage_CL
idp_lookup.json	event join	sonicwall-idp-lookup	event join	event join
session_lookup.json	event join	sonicwall-session-lookup	event join	event join
manifest.json, errors.json, notes.json, local_gate.json, COMPLETE.json	operator evidence, not event inputs	same	same	same

Dedicated containers must contain exactly the intended batch, not a mixture of builds. The ledger's scope_id and snapshot_id identify the local batch; event identity does not require these assessment fields. Complete staged row-set/content comparison detects extra, missing, changed or duplicated rows, including records outside analytical windows. Never filter the comparison down to hunt positives. A wholly omitted tenant/collector remains an independent inventory-audit failure, not something this program discovers.

1. Original event clock versus native selection clock

Original `event_ms` is always a strict non-Boolean integer UTC millisecond time matching independently parsed raw original_ms. Fixed analytical bounds:

- B=1791480600000: October 8, 2026, 10:30 AM MST America/Phoenix.
- S=1793516400000: November 1, 2026, midnight MST.
- E=1794072600000: November 7, 2026, 10:30 AM MST, exclusive.
- H2 VPN [S,E−300000); IdP [S−300000,E). H4 inclusive zero through 3600000 ms after a seed, all events before E; last-hour sessions are right-censored.

The evidence interval is 30 days; the pre-November baseline is 23.5625 days, not known benign. Prior unauthorized access can contaminate novelty. Incomplete retention, ownership changes or suspected earlier access require separate investigation. November 1 is an analyst floor, not an exploitation start. Widening requires coordinated source coverage, settings, all predicates/lookups and validation, never a one-platform edit.

Platform	Native-time requirement and selector
Sentinel	Specified import-time TimeGenerated design, untested and not deployed; original event_ms remains a long. If used, set every event and ledger query time range to the data import-time interval, not historical October dates.
CQL	For events, actual native @timestamp must equal original event_ms. Bounded historical joins depend on this. Ledger native time may be preparation/import time, so use a separate complete selector.
Splunk / Elastic	Record the actual configured native-time policy and full observed bounds. Select the entire batch, whether native time follows original event time or staging time; numeric event_ms predicates determine the analytical windows. SPL earliest=0/latest=now assumes no future native records; if that is false, stop and correct the actual selector/configuration.

Specified Sentinel design: untested and not deployed

This DCR design is untested and not deployed. If used, set the Sentinel query time range to the data import-time interval; `event_ms` selects the historical hunt range. Do not select October and November dates as the native query time range for newly imported data.

Azure Monitor documents the Logs Ingestion API, typed custom tables/DCR streams and the transformation `source | extend TimeGenerated = now()`. This supports staging historical logical events with a current native clock. It does **not** preserve an old TimeGenerated value by exception: Microsoft can replace timestamps older than two days before receipt or more than one day future. We deliberately do not use the original time for that column.

The accompanying sentinel-table.json, sentinel-coverage-table.json and sentinel-dcr.json are unexecuted resource **specifications**, not a deploy script. They define Analytics-plan custom tables, long event/counter columns, string evidence/identity/flag columns, and an explicit DCR projection retaining only the documented logical columns plus current TimeGenerated. Replace resource identifiers only after authorization; preserve existing resources or use dedicated names. A sanctioned identity with ingestion permission, approved workspace/DCR/table configuration, region/network settings and authorized costs are necessary. No account, tenant, credential, resource or budget is supplied by this package.

```
source | extend TimeGenerated = now() | project TimeGenerated, <the explicit logical
columns from the accompanying DCR>
```

The actual JSON contains complete field lists, not this explanatory placeholder. Do not change event_ms to now(), convert it through floating-point seconds or filter out old values during ingestion. JSON payload matches the stream declaration; transformation output matches the destination table schema. The official tutorial describes configuration and permissions; those externally visible/cost-incurring steps need separate approval.

The prepared logical payload exists before upload. After import, export its actual logical columns and native TimeGenerated independently using a custom range that covers the actual import activity. Hash/compare the actual typed logical projection, excluding only native/engine metadata, against the

prepared data. Record native observations separately within the measured load horizon; no original/native equality is required for Sentinel. The logical payload must not change after this observation.

This is **documented feasibility plus reference tests**, not a tested Azure deployment. Without real complete read-back and engine validation, stop interpretation. The same rule applies to every platform; a local test never proves a source archive, actual ingestion or parser behavior.

2. Closed input schema and atomic identity admission

`input.json` has exactly raw, adapter and inventory arrays. The machine-readable CONTRACT.json lists every allowed field and output type. Unlisted input fields are errors, not implicitly trusted context. In particular, adapter session_key, join_key, native_ms, snapshot/readiness values and all joined-helper names are forbidden. Retain extra vendor context separately with original bytes, rather than silently importing arbitrary columns into the hunt table.

Raw receipts and source inventory

Raw objects: coverage_source, raw_id, receipt_id, receipt_ms, original_ms, time_status, raw_sha256, digest_verified, digest_evidence_ref, class_status, class_rule_id, class_evidence_ref. Identities are trimmed nonempty trusted strings; all required times are exact integer UTC ms in [10^12,10^14). Reject booleans, floats, seconds-scale values and nonstandard NaN/Infinity. Targeted records require time_status=validated and valid original_ms. The supplied digest is 64 lowercase hex with independent byte-level verification evidence. The reference compares digests; it does not hash unseen archive bytes or prove an evidence flag truthful.

A receipt is a delivery, distinct from immutable source/raw message identity. Identical copies of a receipt count once; retransmissions with different receipt IDs count separately without becoming new events. Reuse of receipt identity for differing envelopes, differing immutable digest/original-time/classification versions, orphan transformations or multiple distinct transformations for one targeted message reject the entire event batch. This reference supports one canonical event per targeted message. Arbitrary one-to-many expansion needs a separately audited adapter contract, not invented child counts.

Classifications: targeted requires representation; excluded requires an audited rule/evidence and remains in receipt counts; unrecognized rejects the batch and remains visible. An audited exclusion is not a dropped record. Independently review exclusions after parser/firmware changes. False classifier exclusions cannot be disproved by a local program.

Inventory objects: coverage_source, tenant_id, families, family_evidence_refs, raw_expected_count, receipt_start_ms, receipt_end_ms, original_coverage_start_ms, original_coverage_end_ms, watermark_ms, source_active_probe_verified, receipt_evidence_ref, classification_audit_ref, baseline_reviewed, binding_scope_evidence_ref. No redundant required_hunts list is accepted. Family dependencies are fixed: VPN affects H1 through H4, IdP H2, endpoint H4. Each tenant needs

VPN for H1/H3, VPN+IdP for H2 and VPN+endpoint for H4; another tenant cannot supply a missing family.

families is a nonempty unique subset of vpn/idp/endpoint; combined collectors list every actual population. family_evidence_refs maps each family to actual inventory/continuity evidence, including silent/all-excluded populations. Endpoint scope proof is required even with zero rows. Independent raw_expected_count is unique delivery receipts counted **before** classification. Original coverage must include B through a measured watermark at least E; receipt horizon must cover every acquisition time. Active probes, complete sequences/archives and late-arrival/backfill measurement establish coverage, not surviving min/max times or a guessed universal lag. Baseline review addresses coverage/contamination risk, not a claim of benign history. References prove presence only; actual evidence requires independent audit.

Adapter identities and decision fields

Core: coverage_source, raw_id, record_id, tenant_id, kind, event_ms. Kinds: vpn_success, idp_saml_success, endpoint_process, endpoint_admin_logon. Input text columns are strings/null, not arrays/dictionaries or native policy booleans. Output optional missing strings are explicitly empty strings, never null, matching Kusto's string type. Every output event column except event_ms is a string. Required identities are never synthesized from empties. Readback uses this same documented logical representation; inspect actual schemas as well as values. Invalid core identity, unknown kind, mismatched trusted source tenant or composite field rejects the whole batch, even outside the analytical window. No partially eligible survivor is released.

VPN requires primitive appliance_id, session_incarnation and principal_id. IdP requires principal_id/service_id. Endpoint requires endpoint_id; bound endpoints also require appliance_id/session_incarnation. Record collision identity uses **inventory-owned tenant plus record_id**, not the adapter's tenant claim. VPN start collision identity is the trusted tenant/appliance/session-incarnation tuple, with principal/time/source-IP consistency checked across all admitted starts. Conflicting starts reject the batch. Known endpoint activity may share that session without becoming a second VPN start.

Output-only session_key and join_key are generated from primitives using compact UTF-8 JSON tuples [tenant,appliance,session_incarnation] and [tenant,principal,service]. The supplied encoding is never collision identity because it is not accepted as input. Equivalent preparation must produce identical scalar keys and independently test Unicode/escaping round-trips. No shared UNKNOWN placeholders, display-name merging, cross-domain/guest collapse or public-IP-only joins.

Decision fields: - saml_enabled, expected_saml, privileged: true/false/unknown strings with historical policy/privilege evidence. - auth_path: saml/alternate_verified/unknown. alternate_verified is an analyst-defined fact backed by actual method evidence, not a documented SonicWall enum or a guessed list of methods. - *_evidence_ref: actual auth/policy/privilege/session/app/success provenance. A nonblank string alone does not prove truth. - source_ip: independently validated/canonical remote VPN address. The reference checks nonblank only, not IP parsing, proxies or IPv4-mapped IPv6 correctness. - process_name: lower-case executable basename for bound

process events; paths, colon/slash/backslash, upper-case and C0 control characters cannot become ready. DEL and C1 are not rejected by this reference predicate; the seven-name hunt filter is separate. Preserve original path elsewhere. parent_name/command_line are optional actual context, not a process-tree suspicion predicate. - admin_verified: true only with independent administrative-privilege evidence, not any successful logon.

Endpoint binding states: bound requires proved historical device/session relationship, binding_evidence_ref and session_evidence_ref. Use leases, flows and device ownership, accounting for NAT, reused addresses and reconnects; VPN and endpoint principals may differ. not_applicable requires proof of no relevant overlapping session and source/device scope, remains accounted and H4-ineligible, and need not have an H4 basename. Missing session key alone does not establish it. unresolved/missing proof means an H4 gap where relevant; no linker is implemented here.

Actual mapping candidates, **not validated deployment provenance**: Elastic exposes user.name/source.ip/observer fields; sonicwall.firewall.sid is a signature ID and sess a session type, not a verified unique VPN instance. Entra CreatedDateTime/UserId/AppId/resource/ResultType can inform mapping, but success 0 does not prove SAML assertion consumption or MFA challenge; cached assertions may not emit another event. Inspect actual Okta/Falcon schemas rather than inventing eventType fields. MDE Timestamp/DeviceId/FileName/InitiatingProcessFileName/ProcessCommandLine are candidates. Firmware-specific SonicWall success/method codes and sanitized records still require real validation; CNA does not document them.

3. Ledger and local checks

Scope sonicwall-20261107-v6; one ledger row per message plus a sentinel per expected source. Fourteen counters: raw_records, raw_expected_records, targeted_messages, excluded_records, unrecognized_records, normalized_matches, timestamp_gaps, mapping_gaps, manifest_gaps, binding_not_applicable, h1_gap..h4_gap. Source failures and missing families set affected sentinel gaps even with zero events. Structural rejection makes every local gate BLOCKED and emits no events/lookups; retain the original failed inputs and error report.

normalized_matches describes raw-to-adapter correspondence, not successful SIEM ingestion. No generated sink_gaps or universal PASS counter exists. Real native readback is an independent requirement, not a locally asserted zero. Local per-record h*_ready flags only describe supplied decision evidence; their true values do not clear source/engine gates. Unknown relevant auth/policy/privilege/binding evidence creates gaps. H3 history needs usable identity/session/IP, not historical auth/privilege enrichment; recent privileged targets require known policy/privilege. Exact session keys deduplicate H3; H2/H4 may preserve multiple event observations, not unique-session counts.

Local gates are BLOCKED or EXTERNAL_VALIDATION_REQUIRED. The optional check_export(expected, observed, platform, start_ms, end_ms) compares a complete independently exported array of {logical: actual_projected_record, native_ms: actual_observed_integer}; exact

content/types/multiplicity and actual native selection horizon must agree. CQL event exports additionally enforce native/original equality; Sentinel does not. An empty error list is an artifact comparison only, not a real export, source completeness, engine result or approval. Native time is never copied from the proposal to fabricate observation.

4. Join namespaces and Elastic lookups

Every H2/H4 left input explicitly projects only needed logical fields BEFORE joining, so stale physical-table helper columns cannot masquerade as joined evidence. RHS queries also project actual original time/principal before aliases. CQL projections explicitly retain actual @timestamp as native metadata; no joined helper is retained from the left. Joined evidence uses rhs_idp_ms / rhs_vpn_ms / rhs_vpn_principal; KQL therefore cannot accidentally read an unsuffixed left column. This defense is required even with closed adapter JSON because physical table schemas can outlive input changes.

Elastic lookup mode requires a supporting deployed release, exact keyword/long types and all-match content. Minimal schemas:

```
PUT sonicwall-idp-lookup
{"settings":{"index.mode":"lookup"},"mappings":{"dynamic":"strict","properties":
{"join_key":{"type":"keyword"},"rhs_idp_ms":{"type":"long"}}}}
```

```
PUT sonicwall-session-lookup
{"settings":{"index.mode":"lookup"},"mappings":{"dynamic":"strict","properties":
{"session_key":{"type":"keyword"},"rhs_vpn_ms":{"type":"long"},"rhs_vpn_principal":
{"type":"keyword"},"rhs_vpn_saml_enabled":{"type":"keyword"},"rhs_vpn_expected_saml":
{"type":"keyword"},"rhs_vpn_auth_path":{"type":"keyword"}}}}
```

IdP lookup includes EVERY h2-ready success in [S−5m,E), not latest-per-key. Session lookup includes EVERY h4-ready VPN start in [S,E). Distinct row/document IDs preserve all matches, never key-based overwrite. Compare loaded counts, exact full payloads, mappings/fieldcaps and expected source populations with the prepared manifest. No event_ms, tenant_id or other helper columns may be injected into these lookup schemas. Do not mix batches.

5. Safe partitioning and engine completion

Pre-count exact selected left/right populations with readiness/family/time predicates and independent source totals. Verify complete jobs/exports, per-key fan-out and actual versions/caps. A displayed row count is not completion evidence.

- H1: tenant and core original-time partitions; deduplicate overlapping output by tenant/record_id.
- H2: core VPN half-open shards retain ALL same-key IdP context ±300000 ms within global bounds. Do not cut both sides at midnight or partition by bare username/IP. Deduplicate leads by tenant/appliance/record_id. Finer padded shards or stop if still over cap.

- H3: complete tenant/appliance/principal/source-IP history B..E; time shards must merge exact per-session prior/recent indicators before group sums/novelty. Retain 3+ sessions; November-only history cannot establish novelty.
- H4: verified session relationship, not endpoint principal. VPN-time shards need forward one-hour context; endpoint-time shards need preceding one-hour VPN context. Clip at S/E and preserve right-censoring; deduplicate analyst presentation by actual VPN-session/endpoint-event relationship.

Splunk max=0 lifts per-left match restriction, not default RHS row/time limits. CQL limit=200000 bounds total RHS and max=200000 per key; groupBy(limit=max) remains memory/group bounded, both H3 stages require completeness, and historical joins use two passes. ES|QL LIMIT 10000 bounds output, not necessarily earlier aggregate input. KQL has result/memory/partial-failure limits. Unknown completion means inconclusive. No universal in-query truncation signal is supplied.

H4 follows H1 seeds only and seven generic Windows basenames/admin logons, not H2-only leads, renamed tools or all lateral movement. Nearby IdP success is coarse account/app/time correlation, not assertion consumption. Temporal endpoint association is not causation or proof of CVE exploitation.

6. Four-platform independent count readback

Run over the complete actual native-time selection, NOT event_ms-filtered positives. Compare every kind, including expected zeros omitted from grouped output, then exact full logical row content/types/multiplicity and native observations independently.

```
index=sonicwall_hunt earliest=0 latest=now
| stats count AS records BY kind
```

```
FROM sonicwall-hunt
| STATS records=COUNT(*) BY kind
| LIMIT 10000
```

```
dataset="sonicwall_hunt"
| groupBy([kind],function=count(as=records),limit=max)
```

```
SonicWallHunt_CL
| summarize records=count() by kind
```

Ledger: compare complete source set, total rows, exact counters/payloads, scope and batch to the independent inventory and coverage.json. Unexpected scopes/batches or missing sentinel rows fail. Actual ledger native time may be current, including CQL. Lookup content/schema checks remain separate.

```
index=sonicwall_coverage earliest=0 latest=now
| stats count AS ledger_rows BY scope_id snapshot_id coverage_source
```

```
FROM sonicwall-coverage
| STATS ledger_rows=COUNT(*) BY scope_id,snapshot_id,coverage_source
| LIMIT 10000
```

```
dataset="sonicwall_coverage"
|
groupBy([scope_id,snapshot_id,coverage_source],function=count(as=ledger_rows),limit=max)
```

```
SonicWallCoverage_CL
| summarize ledger_rows=count() by scope_id,snapshot_id,coverage_source
```

For query-specific pre-join counts, retain exactly the source/readiness/family/time prefix then count before joins, projections or limits. Include required context/padding for every partition. All code here remains unexecuted in native engines; target schema/version/source/retention/classification/binding/export/job evidence is still required.