

PARTICIPANT GUIDE

Tracing the Unauthorized Path: Evidence-Based Lateral-Movement Reconstruction

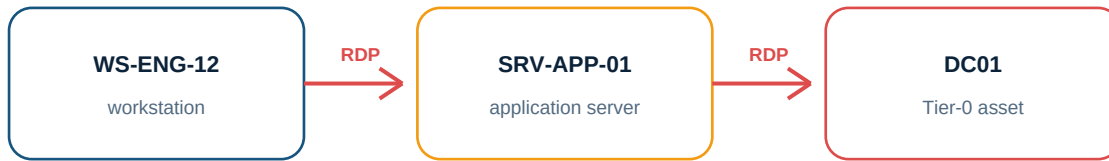
Participant lab guide | Two-hour offline workshop

Desire Abdoul Kader Bonzi | Tetsutaro Uehara

Ritsumeikan University | 19 October 2026 | Singapore

The forensic question

A privileged account may be entitled to administer a critical asset while still using a route that policy does not authorize. This workshop reconstructs the route actually observed and evaluates it separately from entitlement.



Approved route requires JUMP-PAW01 - the observed path bypasses it.

Core question: Who reached what, from where, through which route - and was that observed route authorized?

Learning goals

- Reconstruct an identity-reachability timeline from network and Windows authentication evidence.
- Apply strict flow, host, service, time, and logon-ID joins before asserting a pivot.
- Distinguish identity entitlement from authorization of the observed route.
- Record insufficient evidence or context instead of filling gaps with assumptions.
- Re-run the same case and verify a stable derivation identifier.

Scope

This is a bounded educational proof of concept. It supports controlled RDP/SMB and Windows 4624/4672/4648 evidence, plus an optional CloudTrail AssumeRole extension. It is not a production forensic engine or a real-time detector.

Before the workshop

Extract the ZIP to a writable local folder. This release supports Python 3.11 or 3.12 only; Python 3.13 and later are not supported by the bundled offline wheels. No administrator rights, Elastic Stack, Security Onion, or workshop Internet access is required.

macOS or Linux

```
python3 workshop/scripts/check_python_version.py
python3 -m venv .venv
source .venv/bin/activate
python3 -m pip install -r workshop/requirements-workshop.txt
python3 workshop/scripts/verify_environment.py
```

Windows PowerShell

```
py -3.11 workshop\scripts\check_python_version.py
py -3.11 -m venv .venv
.\.venv\Scripts\python.exe -m pip install -r workshop\requirements-workshop.txt
.\.venv\Scripts\python.exe workshop\scripts\verify_environment.py
```

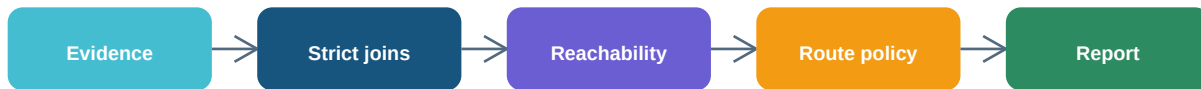
Checkpoint: the preflight ends with WORKSHOP PREFLIGHT: PASS.

Offline installation

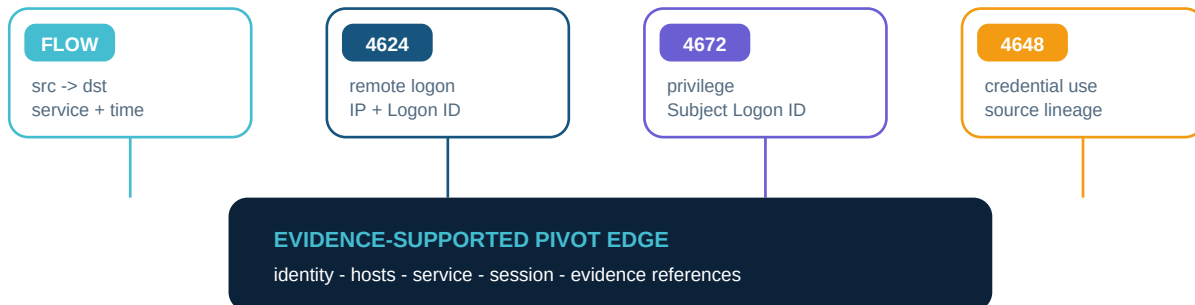
```
python3 -m pip install --no-index --find-links workshop/wheelhouse -r
workshop/requirements-workshop.txt
```

The bundled wheelhouse covers the core dependencies for Python 3.11 and 3.12 on common 64-bit Windows, Intel/Apple Silicon macOS, and x86-64 Linux platforms. If the default interpreter is 3.13 or later, explicitly select Python 3.12 or 3.11. Native EVTX support is optional and not needed for the supplied workshop fixtures.

How the pipeline works



A pivot edge is not a guess



The materializer first requires a compatible network observation and successful 4624 remote logon. Event 4672 confirms privilege only when its user and Subject Logon ID match the 4624 session. Event 4648 confirms a credential transition only when source-session lineage is supported and unambiguous.

No evidence-supported join means no asserted edge. Missing evidence is documented; it is not silently repaired.

Core artifacts

Artifact	Role
PCAP	Observed source, destination, service, and time
4624	Successful remote logon and target session ID
4672	Special privileges for the same logon session
4648	Explicit credential use with supported lineage
Route policy	Approved path and completeness scope

Exercise 1 - reconstruct the observed route

```
python3 workshop/scripts/run_core_case.py --out-dir workshop_output/core_case_a
```

Inspect these outputs

- **results/pivot_edges.json** - Evidence-supported Windows pivot edges
- **results/findings.json** - Route-policy classification
- **results/reconstruction_report.md** - Examiner-readable reconstruction
- **manifest.json** - Input hashes, parameters, outputs, timings, and derivation ID

Questions

- Which two pivot edges were materialized?
- Which evidence references support each edge?
- Why is the route critical even though the identity is entitled to Tier 0?
- Which evidence and context files are hashed in the manifest?

Checkpoint: 2 pivot edges and one CRITICAL_UNAPPROVED_PATH finding.

Interpretation

The result does not mean that privileged administration is always malicious. It means the observed route violates the declared complete policy scope and reaches a critical asset.

Exercise 2A - loose versus strict correlation

The example has two concurrent incoming sessions for the same identity on the pivot host. The outgoing hop names a third source session. A loose same-user/time rule cannot distinguish them.

```
python3 workshop/scripts/compare_join_modes.py
```

Rule	What it uses	Expected pairs
Loose	Host continuity + same user + time	2 unsupported candidates
Strict	The above + source Logon ID lineage	0 asserted pairs

A username is not a session. When no incoming Logon ID supports the outgoing source session, the method rejects the pivot sequence.

Questions

- Which assumption makes the loose rule fabricate two sequences?
- What additional evidence would support one of the paths?
- How should an examiner state the result when that evidence is absent?

Checkpoint

The command prints two loose pairs, zero strict pairs, and ABSTENTION CONTROL: PASS.

Exercise 2B - ambiguous credential transition

Two event 4648 records are plausible for the outgoing authentication, but they name different source sessions. Selecting one would manufacture lineage.

```
python3 workshop/scripts/run_ambiguity_demo.py
```

Observation	Expected result
Plausible 4648 source sessions	2
Credential-transition state	unknown
Evidence-supported pivot sequences	0

The network/authentication edges may still be materialized. The credential transition and multi-hop continuity remain unconfirmed.

Questions

- Why is the transition state unknown rather than confirmed?
- Why are the two source sessions not interchangeable?
- What exact limitation should the examiner record?

Checkpoint

The command ends with AMBIGUITY CONTROL: PASS and asserts no pivot sequence.

Exercise 3 - evaluate three policy states

Evidence and join parameters remain fixed. Only the declared route-policy context changes.

```
python3 workshop/scripts/run_policy_comparison.py
```

Policy state	Edges	Expected result
Complete; observed route prohibited	2	CRITICAL_UNAPPROVED_PATH
Complete; observed route approved	2	EXPECTED
Incomplete for evaluated scope	2	INSUFFICIENT_CONTEXT

A missing allow-list entry becomes prohibition only when the policy declares itself complete and active for the evaluated identity, environment, and target tier.

Questions

- Why do the pivot edges stay identical across all three runs?
- Why is incomplete policy context not the same as a prohibited route?
- Which organizational record would resolve the uncertainty?

Exercise 4 - reproduce the derivation

```
python3 workshop/scripts/run_core_case.py --out-dir workshop_output/core_case_b
```

Print the derivation IDs

```
python3 -c "import json;
print(json.load(open('workshop_output/core_case_a/manifest.json'))['derivation_id'])"
python3 -c "import json;
print(json.load(open('workshop_output/core_case_b/manifest.json'))['derivation_id'])"
```

Checkpoint: the IDs match. Generation time and measured stage timings may differ and are excluded from the stable derivation ID.

A matching ID shows repeatable processing of the declared bundle. It does not prove that the source artifacts were complete, untampered before capture, or interpreted correctly.

Optional hybrid extension

```
python3 workshop/scripts/run_core_case.py --cloud --out-dir workshop_output/hybrid_case
```

Inspect results/reachability_edges_v2.json. Two Windows pivots and one successful AWS AssumeRole transition use a shared edge vocabulary while retaining native evidence references. This does not prove that the Windows and AWS identities are the same person or session.

Final limitations

- A missing edge is not proof that movement did not occur.
- The current materializer is limited to RDP and SMB for Windows pivots.
- The AWS adapter covers successful sts:AssumeRole events, not all cloud actions.
- Batch timings are not a real-time latency claim.
- The controlled fixtures demonstrate functional behavior, not production accuracy or generalization.

Take-away

The method's value is not seeing more than every other system. It is producing a route finding that another examiner can inspect, reproduce, and challenge - with uncertainty left visible.