SEAS-8414 Week 06 Student Lab Guide

SEAS-8414 Week 06 Student Lab Guide: Digital Twin Trustworthiness and Remediation Simulation

Start Here: Download the Student ZIP

Before running any lab commands, download the Week 06 student package from the course site:

https://8414.bwater.io/downloads/labs/packages/seas8414-blackboard-week-06-2026.05.0_4e76a52f_aws8414.zip

The recommended path is the package Makefile:

```
unzip seas8414-blackboard-week-06-2026.05.0_4e76a52f_aws8414.zip
cd seas8414-blackboard-week-06-2026.05.0_4e76a52f_aws8414
make week06
```

The Makefile calls `run-week06-lab.sh`, extracts the nested runtime ZIP, starts the lab, runs the Week 06 API workflow, saves evidence under `lab-results/week-06/evidence/`, generates `lab-results/week-06/index.html`, and cleans up containers at exit.

You can also download the runner directly from https://8414.bwater.io/downloads/labs/scripts/run-week06-lab.sh.

Manual extraction uses two ZIP layers. First extract the weekly Blackboard ZIP, then extract the nested runtime ZIP:

```
unzip seas8414-blackboard-week-06-2026.05.0_4e76a52f_aws8414.zip
cd seas8414-blackboard-week-06-2026.05.0_4e76a52f_aws8414
unzip runtime/seas8414-student-lab-2026.05.0+4e76a52f.aws8414.zip
cd seas8414-student-lab-2026.05.0+4e76a52f.aws8414/student-lab
```

Run the instructions in this guide from that `student-lab/` directory. The matching screencast and LLM prompt are published next to the ZIP on the labs page:

- Screencast MP4: https://8414.bwater.io/downloads/labs/screencasts/phase06-lab-screencast.mp4
- LLM Prompt: https://8414.bwater.io/downloads/labs/prompts/phase06-lab-llm-prompt.md

- Run Script: https://8414.bwater.io/downloads/labs/scripts/run-week06-lab.sh

---

Breakwater Phase 6: Digital Twin Trustworthiness and Remediation Simulation Lab

Simulation Analytics -- What Happens If We Apply This Fix?

Phase 6 introduces the digital twin construction, scenario-based attack simulation, remediation simulation with checkpoint/rollback, cascading failure detection, zero-disruption validation, and twin trustworthiness assessment. These capabilities enable safe remediation by testing every security fix against a twin model before production deployment, while preserving a hard boundary: a digital twin is a decision-support model, not a source of truth.

Within the SEAS-8414 analytics taxonomy (Chapter 6, Section 6.1), this lab is **simulation analytics**: *preview the impact of remediation actions before deployment*. Every remediation plan tested here has been validated against the twin, not against production:

```
Descriptive   (Ch 1, Lab 1)  "What devices exist on the network?"
Diagnostic    (Ch 2, Lab 2)  "What are these devices doing?"
Detective     (Ch 3, Lab 3)  "What vulnerabilities do they have?"
Predictive    (Ch 4, Lab 4)  "What attack paths are likely?"
Prescriptive  (Ch 5, Lab 5)  "What offensive tests should we run?"
Simulation    (Ch 6, this lab)  "What happens if we apply this fix?"
Autonomous    (Ch 7–12)      "Act on it"
```

The analytical tools you will practice include: graph-based dependency analysis (cascade detection), information-theoretic drift measurement (KL-divergence and Jensen-Shannon divergence), constrained optimisation (patch ordering under concurrency limits), Monte Carlo simulation (risk uncertainty quantification), and fidelity testing (twin-to-production correspondence). The output is a validated remediation plan with quantified risk reduction, identified cascading failures, rollback checkpoints, and an explicit authority verdict: trust, trust with limits, resync, rebuild, or block.

What to Expect

What you will do: Run 11 hands-on exercises that require you to build digital twins, design attack scenarios, compare remediation strategies, diagnose cascading failures, design constrained patch orderings, validate remediation plans for zero disruption, and decide whether the twin is trustworthy enough to support a remediation decision. You will use the Breakwater API to execute simulations, but the intellectual work is in the design, comparison, and analysis.

How long it takes: Approximately 5 to 6 hours, including reading time and written analysis. Individual exercises range from 15 to 35 minutes. Several exercises require written deliverables (remediation plans, cascade analysis reports, what-if comparison matrices, and a twin authority assessment).

What you will learn:

- How digital twins are constructed from scan data and why profile mapping accuracy matters
- How attack scenarios establish pre-remediation risk baselines for before/after comparison
- How cascading failures propagate through service dependencies and how to predict them
- How to design remediation plans that respect concurrency constraints and downtime windows
- How to compare multiple remediation strategies using quantitative risk metrics
- How to validate that a remediation plan achieves zero disruption before production deployment
- How to determine when twin drift, low fidelity, or missing evidence should block a remediation recommendation

Prerequisites:

- Docker and Docker Compose installed
- `jq` installed (`brew install jq` on macOS, `apt install jq` on Linux)
- `curl` installed
- Completed Phase 1 through Phase 5 labs
- **Chapter 6 of the textbook read**, particularly Sections 6.1 (Analytics Context), 6.3 (Twin Construction), 6.5 (Scenario Engine), 6.6 (Remediation Simulation), and 6.10 (Cascading Failure Detection)
- The Breakwater student lab environment running

Lab Environment Setup

```
cd student-lab/
docker compose up -d
sleep 30
docker compose ps
```

Register or authenticate and prepare scan data:

> Use `student@example.com`. The API validator rejects `.local` as a special-use domain, so the labs standardize on an RFC 2606 example domain.

```
TOKEN=$(curl -s -X POST http://localhost:8100/v1/auth/register \
  -H "Content-Type: application/json" \
  -d '{"email":"student@example.com","password":"SecurePass!2026","full_name":"Lab
        Student"}' \
  | jq -r '.access_token // empty')
if [ -z "$TOKEN" ]; then
  TOKEN=$(curl -s -X POST http://localhost:8100/v1/auth/login \
    -H "Content-Type: application/json" \
    -d '{"email":"student@example.com","password":"SecurePass!2026"}'
        \
```

```
    | jq -r '.access_token')
fi

SCAN_ID=$(curl -s "http://localhost:8100/v1/scanning/smart-
        scan/history?limit=1" \
  -H "Authorization: Bearer $TOKEN" | jq -r '.scans[0].scan_id')

echo "Token: ${TOKEN:0:20}..."
echo "Scan ID: $SCAN_ID"
```

Deploy the digital twin:

```
TWIN_ID=$(curl -s -X POST
        "http://localhost:8100/v1/twin/deploy/$SCAN_ID" \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{"auto_scenario": false}' | jq -r '.data.twin_id')

echo "Twin ID: $TWIN_ID"
```

Required Decision Gate: Twin Authority

Before recommending any remediation plan, you must decide whether the twin is authoritative enough for that specific decision. A twin may be good enough to compare broad remediation strategies and still not be good enough to authorize a change against a critical PLC, HMI, camera, or gateway.

Use this gate before Exercise 9 and update it again after Exercise 11.

Metric	Required Evidence	Block Condition
Device coverage	Scan device count vs. twin device count	Missing critical assets or unknown targets
Profile confidence	Mean and minimum confidence for mapped devices	Remediation target mapped to low-confidence fallback
Fallback profiles	Count and location of fallback mappings	Fallback on device in the remediation path
Service drift	Latest observed ports/services vs. twin services	Changed service on target, dependency, or safety-critical path
Topology drift	New/removed device, subnet, route, or dependency	Unmodeled dependency or segmentation edge
Traffic replay fidelity	Correct replay responses / total replay responses	Below 80% for affected protocol
KL divergence	Behavioral drift by metric	Severe unexplained drift on affected device class
Jensen-Shannon divergence	Bounded drift score for stable comparison	Medium or high drift on a critical asset
Cascade prediction	Expected vs. observed dependency behavior	Unexplained cascade mismatch

Metric	Required Evidence	Block Condition
Freshness	Time since scan and last rebuild event	Exceeds freshness policy or follows a major change event

Your authority verdict must be one of:

- **Trust:** The twin supports this remediation decision.
- **Trust with limits:** The twin supports only part of this decision; state the boundary.
- **Resync:** Fresh scan, traffic replay, or dependency evidence is required before recommendation.
- **Rebuild:** The twin is stale or structurally invalid.
- **Block:** The simulation must not be used to approve the remediation.

Exercise 1: Evaluate Twin Fidelity Through Profile Mapping Analysis

Time: ~25 minutes | **Difficulty:** Intermediate | **Textbook:** Section 6.3

Context

The digital twin is only useful if it accurately represents the real network. Profile mapping translates real devices into simulation profiles. Inaccurate mapping means the twin cannot predict production behavior. This exercise requires you to evaluate the mapping quality and identify gaps.

Task

Part A: Examine the twin topology and profile assignments:

```
# Twin status
curl -s "http://localhost:8100/v1/twin/$TWIN_ID/status" \
  -H "Authorization: Bearer $TOKEN" | jq .

# Device-to-profile mapping
curl -s "http://localhost:8100/v1/twin/scan/$SCAN_ID" \
  -H "Authorization: Bearer $TOKEN" \
  | jq '[.data.topology_data.devices[] | {device_ip, profile_name,
        device_type, mapped_ports}]'
```

Part B: Compare twin device count to scan device count:

```
# Scan device count
echo "Scan devices:"
curl -s http://localhost:8100/v1/scanning/smart-scan/$SCAN_ID/results
        \
  -H "Authorization: Bearer $TOKEN" | jq '.hosts | length'

# Twin device count
echo "Twin devices:"
```

```
curl -s "http://localhost:8100/v1/twin/scan/$SCAN_ID" \
  -H "Authorization: Bearer $TOKEN" | jq '.data.topology_data.devices
        | length'
```

Part C: Count devices per profile:

```
curl -s "http://localhost:8100/v1/twin/scan/$SCAN_ID" \
  -H "Authorization: Bearer $TOKEN" \
  | jq '[.data.topology_data.devices[].profile_name] | group_by(.) |
        map({profile: .[0], count: length})'
```

Analysis Questions

1. **Coverage gap analysis.** Compare the twin device count to the scan device count. If they differ, which devices were not included in the twin? Investigate: are they devices with unrecognized device types? Devices with no open ports? What impact does excluding these devices have on the twin's ability to predict remediation outcomes?

2. **Profile accuracy assessment.** The profile mapper uses three tiers: device type lookup, port inference, and service name hinting. A camera mapped to "camera-variant" with high confidence (0.9) will behave realistically in the twin. A router mapped to "http-debug" with low confidence (0.1) may not. Identify any devices mapped to a fallback profile (http-debug with confidence < 0.3). For each, propose what profile SHOULD have been assigned and what additional scan data would improve the mapping.

3. **Fidelity implications.** If 20% of devices are mapped with low confidence, any remediation simulation that affects those devices will produce unreliable results. Design a "fidelity score" for the twin: `fidelity = sum(device_confidence) / device_count`. Calculate it for the lab twin. At what fidelity score would you refuse to trust the twin's remediation predictions?

4. **Twin vs. reality divergence.** The twin is constructed from scan data at a point in time. If the real network changes (new device deployed, firmware updated, port opened) between the scan and the remediation, the twin becomes stale. How often should the twin be rebuilt? What event should trigger an immediate rebuild? Design a freshness policy.

Exercise 2: Design and Execute an Attack Scenario

Time: ~25 minutes | **Difficulty:** Intermediate | **Textbook:** Section 6.5

Context

Attack scenarios replay adversary techniques against the twin to establish pre-remediation baselines. Without a baseline, you cannot measure whether remediation actually reduced risk. The scenario engine uses deterministic hashing to ensure reproducible results.

Task

Part A: Design a multi-step attack scenario targeting the most valuable assets:

```
# Design your scenario based on Phase 4 attack path analysis
curl -s -X POST "http://localhost:8100/v1/twin/$TWIN_ID/scenario" \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "scenario_type": "attack_replay",
    "steps": [
      {"target_ip": "172.30.0.10", "technique":
        "telnet_default_creds", "severity": "critical"},
      {"target_ip": "172.30.0.11", "technique": "rtsp_unauth",
        "severity": "high"},
      {"target_ip": "172.30.0.20", "technique": "http_exploit",
        "severity": "medium"},
      {"target_ip": "172.30.0.30", "technique": "mqtt_unauth",
        "severity": "high"},
      {"target_ip": "172.30.0.40", "technique": "lateral_movement",
        "severity": "medium"}
    ]
  }' | jq .
```

Part B: Run the same scenario a second time to verify determinism:

```
# Second execution (should produce identical results)
curl -s -X POST "http://localhost:8100/v1/twin/$TWIN_ID/scenario" \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "scenario_type": "attack_replay",
    "steps": [
      {"target_ip": "172.30.0.10", "technique":
        "telnet_default_creds", "severity": "critical"},
      {"target_ip": "172.30.0.11", "technique": "rtsp_unauth",
        "severity": "high"},
      {"target_ip": "172.30.0.20", "technique": "http_exploit",
        "severity": "medium"},
      {"target_ip": "172.30.0.30", "technique": "mqtt_unauth",
        "severity": "high"},
      {"target_ip": "172.30.0.40", "technique": "lateral_movement",
        "severity": "medium"}
    ]
  }' | jq '{devices_compromised: .data.devices_compromised,
        blast_radius: .data.blast_radius, steps_succeeded:
        .data.steps_succeeded}'
```

Analysis Questions

1. **Scenario design rationale.** Explain why you chose these five targets and techniques. Reference the Phase 4 attack graph and Phase 5 pentest results. If the scenario does not include the easiest attack path from Phase 4, modify it and re-run. Does including the easiest path change the blast radius?

2. **Determinism verification.** Compare the two scenario executions. Are the results identical? If yes, explain why determinism is critical for before/after remediation comparison. If the scenario were stochastic (random success/failure), how would you establish a reliable baseline?

3. **Blast radius interpretation.** The blast radius should exceed the directly compromised device count because it includes devices reachable through firewall rules from compromised hosts. Calculate: `amplification = blast_radius / devices_compromised`. An amplification of 3.0 means each compromised device exposes 3 additional devices. What does a high amplification tell you about network segmentation?

4. **Scenario coverage assessment.** Your 5-step scenario tests only 5 of 22 devices. Design a comprehensive scenario that covers ALL device types in the network (at least one camera, one NAS, one router, one sensor, one smart plug). Why is device-type diversity important for baseline completeness?

Exercise 3: Compare Three Remediation Strategies Head-to-Head

Time: ~30 minutes | **Difficulty:** Advanced | **Textbook:** Sections 6.6, 6.8

Context

This exercise is the analytical core of Phase 6. You will design three competing remediation strategies, execute each against the twin, and recommend one based on quantitative comparison.

Task

Part A: Execute three strategies:

```
# Strategy A: Patch vulnerabilities only (no topology changes)
echo "=== Strategy A: Patch Only ==="
curl -s -X POST "http://localhost:8100/v1/twin/$TWIN_ID/remediate" \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "actions": [
      {"action_type": "patch_cve", "target_ip": "172.30.0.10",
        "parameters": {"cve_id": "CVE-2023-25135"}},
      {"action_type": "patch_cve", "target_ip": "172.30.0.11",
        "parameters": {"cve_id": "CVE-2023-25135"}}
    ]
  }' | jq '{strategy: "A_patch", risk_delta: .data.risk_delta,
        cascades: (.data.cascading_failures | length),
        zero_disruption: .data.zero_disruption}'

# Strategy B: Disable insecure services (reduce attack surface)
echo "=== Strategy B: Service Hardening ==="
curl -s -X POST "http://localhost:8100/v1/twin/$TWIN_ID/remediate" \
  -H "Authorization: Bearer $TOKEN" \
```

```
  -H "Content-Type: application/json" \
  -d '{
    "actions": [
      {"action_type": "disable_service", "target_ip": "172.30.0.10",
        "parameters": {"port": 23}},
      {"action_type": "disable_service", "target_ip": "172.30.0.11",
        "parameters": {"port": 554}}
    ]
  }' | jq '{strategy: "B_harden", risk_delta: .data.risk_delta,
        cascades: (.data.cascading_failures | length),
        zero_disruption: .data.zero_disruption}'

# Strategy C: Network segmentation (isolate device classes)
echo "=== Strategy C: Segmentation ==="
curl -s -X POST "http://localhost:8100/v1/twin/$TWIN_ID/remediate" \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "actions": [
      {"action_type": "segment_network", "target_ip": "172.30.0.0/28",
        "parameters": {"source_subnet": "172.30.0.0/28",
        "target_subnet": "172.30.0.16/28"}}
    ]
  }' | jq '{strategy: "C_segment", risk_delta: .data.risk_delta,
        cascades: (.data.cascading_failures | length),
        zero_disruption: .data.zero_disruption}'
```

Part B: Record results in a decision matrix:

Criterion	Strategy A (Patch)	Strategy B (Harden)	Strategy C (Segment)
Risk delta	?	?	?
Cascading failures	?	?	?
Zero disruption?	?	?	?
Implementation cost	Low ($)	Medium ($$)	High ($$$)
Downtime required	Moderate	Low	High
Recurring effort	Per-patch	One-time	One-time

Analysis Questions

1. **Multi-criteria decision analysis.** Using the decision matrix, assign weights to each criterion (risk_delta: 30%, cascades: 25%, zero_disruption: 20%, cost: 15%, downtime: 10%) and calculate a weighted score for each strategy. Which strategy wins? Does the winner change if you shift weights to prioritise zero_disruption (40%)?

2. **Strategy interaction.** If you could combine two strategies, which pair would you choose? Run the combined simulation and compare the risk_delta to the sum of the individual deltas. Is the combined effect additive, superadditive, or subadditive? Explain why.

3. **Cascading failure diagnosis.** Strategy B (disabling services) and Strategy C (segmentation) may produce cascading failures. For each cascade, trace the dependency chain: which device depends on the disabled service or blocked traffic flow? Propose a modification to the strategy that preserves the security benefit while eliminating the cascade.

4. **Reversibility analysis.** Patching (A) is hard to reverse (firmware downgrade is risky). Disabling services (B) is easily reversible (re-enable the port). Segmentation (C) is reversible but complex (remove firewall rules). For a hospital network with zero tolerance for downtime, which strategy's reversibility makes it the safest first step? How does reversibility affect your recommendation?

Exercise 4: Diagnose and Mitigate Cascading Failures

Time: ~25 minutes | **Difficulty:** Advanced | **Textbook:** Section 6.10

Context

Cascading failures occur when a remediation action on device A disrupts device B, which depends on A's services. The twin's dependency graph reveals these cascades BEFORE you touch production. This exercise requires you to provoke, diagnose, and mitigate cascading failures.

Task

Part A: Examine the dependency graph:

```
# Dependency graph
curl -s "http://localhost:8100/v1/twin/$TWIN_ID/cascades" \
  -H "Authorization: Bearer $TOKEN" \
  | jq '{dependency_count: (.data.dependencies | keys | length),
        cascade_analysis: [.data.cascade_analysis[] | {device_ip,
        blast_count: .blast_radius.blast_count, severity:
        .blast_radius.severity}] | sort_by(-.blast_count) | .[:5]}'
```

Part B: Simulate credential rotation on a device with dependents:

```
curl -s -X POST "http://localhost:8100/v1/twin/$TWIN_ID/remediate" \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "actions": [
      {"action_type": "rotate_credentials", "target_ip":
        "172.30.0.30", "parameters": {"service": "mqtt"}}
    ]
  }' | jq '{risk_delta: .data.risk_delta, cascades:
        [.data.cascading_failures[] | {cascade_type, severity,
        description, affected_devices: .affected_devices}]}'
```

Analysis Questions

1. **Dependency graph interpretation.** How many service dependencies exist? Which device has the highest blast count? If that device goes offline, how many other devices lose functionality? For a hospital, identify which dependencies are safety-critical (e.g., a PLC depending on a historian for setpoint data).

2. **Cascade type classification.** The textbook describes three cascade types: service dependency (device B depends on device A's service), segmentation (blocking traffic disrupts legitimate flows), and credential (rotating shared credentials breaks authentication for dependent devices). Which type did the MQTT credential rotation trigger? Why?

3. **Cascade mitigation strategy.** For each cascading failure you observed, propose a mitigation: (a) if it is a credential cascade, rotate credentials on ALL same-profile devices simultaneously; (b) if it is a service dependency, update dependent devices' configuration to use the new service endpoint before rotating; (c) if it is a segmentation cascade, create explicit allow rules for verified dependencies before applying the segmentation rule.

4. **Pre-flight checklist design.** Design a "pre-flight checklist" that a security engineer must complete before executing any remediation action on the production network. The checklist should include: (1) twin simulation results reviewed, (2) cascading failures identified and mitigated, (3) rollback procedure documented, (4) monitoring alerts configured, (5) change advisory board approval obtained. Add two more items specific to IoT/OT environments.

Exercise 5: Design a Constrained Patch Ordering

Time: ~30 minutes | **Difficulty:** Advanced | **Textbook:** Section 6.7

Context

Patching 22 devices simultaneously is risky -- if the patch causes a regression, ALL devices fail at once. Patching one at a time is safe but slow. The constrained patch ordering problem finds the optimal group size and sequencing that minimises total time while respecting a concurrency limit and dependency constraints.

Task

Part A: Identify all devices requiring patching and design groups:

```
# All twin devices
DEVICES=$(curl -s "http://localhost:8100/v1/twin/scan/$SCAN_ID" \
  -H "Authorization: Bearer $TOKEN" | jq -r
        '.data.topology_data.devices[].device_ip')

DEVICE_COUNT=$(echo "$DEVICES" | wc -l | tr -d ' ')
echo "Total devices to patch: $DEVICE_COUNT"
```

Part B: Simulate patching in groups of 3 (concurrency limit = 3):

```
# Group 1: first 3 devices
GROUP1_IPS=$(echo "$DEVICES" | head -3)
ACTIONS=""
for ip in $GROUP1_IPS; do
  ACTIONS="$ACTIONS{\"action_type\": \"patch_cve\", \"target_ip\":
        \"$ip\", \"parameters\": {\"cve_id\": \"CVE-2023-25135\"}},"
done
ACTIONS="[${ACTIONS%,}]"

curl -s -X POST "http://localhost:8100/v1/twin/$TWIN_ID/remediate" \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d "{\"actions\": $ACTIONS}" \
  | jq '{group: 1, actions_applied: .data.actions_applied, risk_delta:
        .data.risk_delta, cascades: (.data.cascading_failures |
        length), zero_disruption: .data.zero_disruption}'
```

Analysis Questions

1. **Group count and total time.** With a concurrency limit of 3 and N devices, you need ceiling(N/3) groups. If each group takes 120 seconds (reboot time), calculate the total sequential patching time vs. the grouped patching time. What is the speedup factor? For 100 devices: sequential = 100 * 120s = 12,000s (3.3 hours), grouped = ceiling(100/3) * 120s = 4,080s (1.1 hours). Speedup = 2.9x.

2. **Dependency-aware grouping.** Devices that depend on each other should NOT be in the same group (if device A depends on device B's service, patching both simultaneously could cause A to fail when B reboots). Using the cascade analysis from Exercise 4, identify which devices should be in DIFFERENT groups. Redesign your groups to respect these constraints.

3. **Wave-based deployment strategy.** Instead of patching all devices at once, patch 20% (wave 1), monitor for 24 hours, then patch the next 20% (wave 2), and so on. This limits blast radius if the patch causes a regression. Design a 5-wave schedule for the lab network, assigning devices to waves based on: (a) criticality (least critical devices first), (b) device type diversity (each wave includes mixed types to detect type-specific regressions), (c) dependency independence (no two interdependent devices in the same wave).

4. **Rollback planning.** For each patch group, design a rollback procedure: what triggers rollback (specific error conditions, health check failures, user reports), who authorises rollback, and how long the rollback takes. Should rollback be automatic or manual? For a camera firmware rollback: the previous firmware must be cached locally because downloading it from the vendor takes 10 minutes.

Exercise 6: Validate Zero-Disruption Remediation

Time: ~25 minutes | **Difficulty:** Intermediate | **Textbook:** Section 6.12

Context

A "zero-disruption" remediation is one that does not cause any cascading failures or service interruptions. The twin validates this by checking whether the post-remediation state has any new service disruptions compared to the pre-remediation state.

Task

Part A: Find a remediation action that achieves zero disruption:

```
# Test: patch a single device with no dependents
curl -s -X POST "http://localhost:8100/v1/twin/$TWIN_ID/remediate" \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "actions": [
      {"action_type": "patch_cve", "target_ip": "172.30.0.10",
        "parameters": {"cve_id": "CVE-2023-25135"}}
    ]
  }' | jq '{zero_disruption: .data.zero_disruption, cascades:
        (.data.cascading_failures | length), risk_delta:
        .data.risk_delta}'
```

Part B: Test a remediation that is likely to cause disruption:

```
# Test: disable a service that other devices might depend on
curl -s -X POST "http://localhost:8100/v1/twin/$TWIN_ID/remediate" \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "actions": [
      {"action_type": "disable_service", "target_ip": "172.30.0.30",
        "parameters": {"port": 1883}}
    ]
  }' | jq '{zero_disruption: .data.zero_disruption, cascades:
        (.data.cascading_failures | length), risk_delta:
        .data.risk_delta}'
```

Analysis Questions

1. **Zero-disruption threshold.** The twin reports zero_disruption = true if there are no cascading failures. This is not sufficient for a doctoral remediation claim. A zero-disruption claim requires: (a) no new cascading failures, (b) no loss of a required service path, (c) no critical dependency broken, (d) no service availability drop below the required threshold, (e) no removal of redundancy without approval, (f) no unexplained traffic replay mismatch, (g) rollback tested or explicitly documented, and (h) a Twin Authority verdict of Trust or Trust with limits. Apply this definition to Part A and Part B. Which claim survives?

2. **Disruption tolerance spectrum.** Not all disruptions are equal. A 5-second MQTT reconnection is acceptable. A 30-minute camera outage during a security incident is not. Design a disruption tolerance matrix for different device types and disruption durations.

3. **Pre-production validation workflow.** Design a complete workflow for validating a remediation action before production deployment: (1) Run simulation on twin, (2) Verify zero-disruption, (3) If cascades detected, modify the plan and re-simulate, (4) Obtain approval, (5) Deploy to one device (canary), (6) Monitor canary for 1 hour, (7) Deploy to remaining devices in waves. What monitoring metrics would you check during the canary deployment?

4. **Twin accuracy under remediation.** The twin's prediction accuracy degrades as you apply more remediation actions because each action changes the twin state further from the real network. After applying 5 remediation actions to the twin without applying them to production, how much confidence do you have in the twin's 6th prediction? Propose a policy: "Rebuild the twin after every N remediation actions or every M days, whichever comes first."

Exercise 7: Analyse Traffic Replay Fidelity

Time: ~20 minutes | **Difficulty:** Intermediate | **Textbook:** Section 6.14

Context

Traffic replay sends real protocol transcripts against the twin and compares simulated responses to expected responses. High fidelity means the twin behaves like the real network; low fidelity means remediation predictions may be inaccurate.

Task

```
# HTTP traffic replay
curl -s -X POST "http://localhost:8100/v1/twin/$TWIN_ID/replay-
        traffic" \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "transcript": [
      {"direction": "send", "data": "GET / HTTP/1.1\r\nHost:
        172.30.0.10\r\n\r\n"},
      {"direction": "recv", "data": "HTTP/1.1 200 OK\r\n"},
      {"direction": "send", "data": "GET /admin HTTP/1.1\r\nHost:
        172.30.0.10\r\n\r\n"},
      {"direction": "recv", "data": "HTTP/1.1 403 Forbidden\r\n"}
    ],
    "target_ip": "172.30.0.10",
    "target_port": 80
  }' | jq '.data | {sent_count, recv_count, duration_ms,
        simulated_responses: (.simulated_responses | length)}'

# RTSP traffic replay
curl -s -X POST "http://localhost:8100/v1/twin/$TWIN_ID/replay-
        traffic" \
  -H "Authorization: Bearer $TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "transcript": [
```

```
      {"direction": "send", "data": "OPTIONS
        rtsp://172.30.0.11/stream1 RTSP/1.0\r\n"},
      {"direction": "send", "data": "DESCRIBE
        rtsp://172.30.0.11/stream1 RTSP/1.0\r\n"}
    ],
    "target_ip": "172.30.0.11",
    "target_port": 554
  }' | jq '.data | {sent_count, recv_count, duration_ms}'
```

Analysis Questions

1. **Fidelity measurement.** Compare the twin's simulated HTTP response to the expected response (200 OK for /, 403 for /admin). If the twin returns 200 for /admin (no authentication enforcement), the twin does not accurately model the device's access controls. What is the fidelity score (correct_responses / total_responses) for your test?

2. **Remediation verification via replay.** After disabling Telnet on a device, you should replay a Telnet transcript against the twin and verify that the service is no longer accessible. Design a verification test suite with 3 transcripts: one for a patched CVE (expected: exploit fails), one for a disabled service (expected: connection refused), one for a rotated credential (expected: authentication failure).

3. **Protocol coverage limitations.** The traffic replay engine generates deterministic simulated responses. It does not understand protocol semantics (it cannot parse RTSP commands or HTTP headers). How does this limit the twin's ability to predict the impact of protocol-level remediations (e.g., disabling RTSP DESCRIBE vs. disabling RTSP entirely)?

4. **Twin fidelity improvement.** If the twin's fidelity score is 60% (6 of 10 simulated responses match real responses), what steps would you take to improve it? Consider: more detailed device profiles, real traffic capture for baseline comparison, protocol-specific simulation engines.

Exercise 8: Design a Risk Comparison Dashboard

Time: ~25 minutes | **Difficulty:** Advanced | **Textbook:** Section 6.8

Context

The risk comparison endpoint aggregates results from all remediation simulations run against a twin, enabling side-by-side comparison. This exercise requires you to design and populate a comparison dashboard.

Task

Part A: Run at least 3 different remediation plans and retrieve the comparison:

```
# Check the risk comparison (aggregates all previous simulations)
curl -s "http://localhost:8100/v1/twin/$TWIN_ID/risk-comparison" \
  -H "Authorization: Bearer $TOKEN" | jq .
```

Analysis Questions

1. **Dashboard design.** Design a visual dashboard layout that presents the risk comparison data to a CISO. Include: (a) a bar chart comparing risk_delta across strategies, (b) a scatter plot of risk_delta vs. cost (Pareto frontier), (c) a table of cascading failures by strategy, (d) a traffic light indicator for zero_disruption (green/red). Sketch or describe the layout.

2. **Pareto optimality.** In multi-objective optimisation, a Pareto-optimal strategy is one where no other strategy is better on ALL criteria simultaneously. From your three strategies, identify which are Pareto-optimal (e.g., Strategy A has the best risk_delta but worst cost; Strategy B has moderate risk_delta and moderate cost -- B is Pareto-optimal if no strategy beats it on both criteria). Which strategies are dominated (worse on all criteria)?

3. **Sensitivity analysis.** The risk comparison depends on the BRS weights used during simulation. If the CISO changes the vulnerability weight from 0.20 to 0.40, would Strategy A (patch-only, which reduces vulnerability) move up in the ranking? Predict the ranking change without re-running the simulation.

4. **Recommendation memorandum.** Write a two-paragraph recommendation memo to the CISO: Paragraph 1 states which strategy you recommend and why (reference the quantitative comparison). Paragraph 2 acknowledges risks and limitations (twin fidelity, cascade uncertainties, model assumptions).

Exercise 9: Build an End-to-End Remediation Pipeline

Time: ~30 minutes | **Difficulty:** Advanced | **Textbook:** Sections 6.6, 6.7, 6.10, 6.12

Context

This exercise synthesises all Phase 6 concepts into a complete remediation pipeline: design, simulate, validate, schedule, and decide whether the twin has authority to support the recommendation.

Written Deliverable (Required)

Produce a **final remediation simulation report** for the lab network that includes:

1. **Baseline twin summary**: device count, profile mapping quality, dependencies, and known limits.

2. **Twin fidelity score**: coverage, confidence, fallback mappings, and freshness policy.

3. **Attack scenario baseline** (from Exercise 2): scenario design, determinism check, blast radius, and coverage gap.

4. **Strategy selection** (from Exercise 3): Which strategy and why.

5. **Cascade mitigation** (from Exercise 4): For each identified cascade, the specific mitigation action.

6. **Patch ordering** (from Exercise 5): Devices grouped into waves with concurrency limit of 3, respecting dependency constraints.

7. **Zero-disruption validation** (from Exercise 6): Which actions achieved the stronger zero-disruption definition, which did not, and how the non-zero actions were modified.

8. **Verification tests** (from Exercise 7): Traffic replay transcripts that confirm each remediation action was applied correctly.

9. **Risk comparison** (from Exercise 8): Quantitative justification for the chosen strategy vs. alternatives.

10. **Twin Authority Gate verdict**: Trust, Trust with limits, Resync, Rebuild, or Block. State what evidence supports the verdict.

11. **Explicit do-not-execute conditions**: Conditions that must stop production deployment even if the simulation looks favorable.

12. **Schedule**: Week-by-week implementation timeline.

13. **Rollback plan**: For each wave, the trigger conditions, authorizer, expected duration, and rollback evidence.

14. **Residual risks and assumptions**: What the twin did not model and what production evidence is still required.

Analysis Questions

1. **Pipeline automation potential.** Which steps of the pipeline could be fully automated (no human in the loop)? Which require human judgment? The twin simulation can be automated. The cascade mitigation design requires human analysis. The patch deployment can be automated with approval gates. Design an automation architecture showing which steps are automated and where humans approve.

2. **Continuous remediation.** Instead of a one-time plan, design a continuous remediation pipeline that: (a) rebuilds the twin after each scan (weekly), (b) automatically simulates the top 3 remediation actions, (c) presents results to the security team for approval, (d) deploys approved actions via the patch management system. What is the expected steady-state reduction in BRS over 6 months of continuous operation?

3. **Twin drift detection.** After deploying remediation actions to production, the twin should be updated to match. But if the production deployment fails silently (e.g., firmware update was applied but did not take effect), the twin and production diverge. Design a drift detection mechanism: what health checks would you run post-deployment to verify twin-production correspondence?

Exercise 10: Design a Digital Twin Strategy for a Multi-Site Enterprise

Time: ~30 minutes | **Difficulty:** Advanced | **Textbook:** Sections 6.1-6.14

Scenario

You are the security architect for a manufacturing company with:

- **Site A (headquarters)**: 500 IT devices, 50 IoT devices (cameras, printers, HVAC)
- **Site B (factory)**: 200 PLCs, 100 HMIs, 50 cameras, 30 network devices
- **Site C (warehouse)**: 20 IP cameras, 10 access control panels, 5 network devices
- Sites are connected via MPLS WAN with segmented routing.
- Budget: $100,000/year for security tooling.
- Staff: 3 security engineers.

Written Deliverable (Required)

Produce a **digital twin deployment strategy** that includes:

1. **Twin architecture**: One twin per site or one enterprise-wide twin? Justify.
2. **Scan cadence**: How often to rebuild each twin. Consider that Site B (factory) changes rarely while Site A (HQ) changes weekly.
3. **Simulation priority**: Which site gets remediation simulation first? (Hint: factory PLCs have the highest consequence of compromise.)
4. **Fidelity requirements**: What minimum fidelity score is acceptable for each site? Factory PLCs require higher fidelity than HQ cameras.
5. **Team allocation**: How the 3 engineers divide their time across sites.
6. **ROI justification**: How digital twin simulation saves money vs. testing remediations directly on production (reference: average cost of a production outage at Site B = $500,000/hour).

Analysis Questions

1. **Scale challenges.** A twin for Site B (380 devices) will have a much larger attack graph than the lab twin (22 devices). How does computational complexity scale with device count for: (a) graph construction, (b) attack path computation (Yen's algorithm), (c) BRS scoring? Is the scaling linear, quadratic, or exponential?

2. **Cross-site attack paths.** If the MPLS WAN allows limited traffic between sites, the enterprise-wide attack graph includes cross-site edges. An attacker who compromises a camera at Site C could potentially reach a PLC at Site B. Should the digital twin model cross-site paths? What is the risk of NOT modeling them?

3. **Twin maintenance cost.** Each twin rebuild requires a full scan plus graph construction. Estimate the annual cost of maintaining 3 twins at weekly rebuild frequency. Include: scan time (30 min/site), twin construction (5 min/site), simulation runs (10 min/site), analyst review (30 min/site per week). Total annual hours = 52 * (30+5+10+30) * 3 / 60 = ?. Does this fit within the 3-engineer staffing budget?

4. **Regulatory justification.** The factory is subject to IEC 62443 (industrial cybersecurity). Section 4-2-3 requires "verification and validation of security countermeasures before deployment." Does digital twin simulation satisfy this requirement? Draft a one-paragraph justification for the auditor.

Exercise 11: Twin Authority Under Drift

Time: ~35 minutes | **Difficulty:** Advanced | **Textbook:** Sections 6.3, 6.12, 6.14

Context

The twin is useful only while it remains close enough to the environment it represents. Aggregate drift metrics can miss small but operationally dangerous changes: one firewall rule, one changed dependency, one newly exposed service, or one critical device remapped to a fallback profile. This exercise requires you to test that failure mode directly.

Task

Construct a drift table using evidence from the lab API outputs you already collected. You may use the actual lab outputs, manually constructed scenario rows, or a small spreadsheet. The goal is not to mutate production. The goal is to reason explicitly about when a model loses decision authority.

Use these controlled drift scenarios:

1. One unknown device is added to the scan but not represented in the twin.
2. Three unknown devices are added to the scan but not represented in the twin.
3. One firewall rule changes on a remediation path.
4. One new service opens on a remediation target.
5. One PLC/HMI or MQTT dependency changes.
6. One high-confidence profile is replaced by a fallback profile.
7. Traffic replay returns a response mismatch for an affected protocol.
8. One subnet is split or one routing edge is removed.

For each scenario, fill this table:

Scenario	Operationally material?	Metric that detects it	KL detected it?	JS detected it?	Authority verdict	Required action
One unknown device	?	?	?	?	?	?
Three unknown devices	?	?	?	?	?	?
Firewall rule changed	?	?	?	?	?	?
New service opened	?	?	?	?	?	?
Dependency changed	?	?	?	?	?	?
Fallback profile substituted	?	?	?	?	?	?

Scenario	Operationally material?	Metric that detects it	KL detected it?	JS detected it?	Authority verdict	Required action
Replay mismatch	?	?	?	?	?	?
Subnet or route changed	?	?	?	?	?	?

Divergence Calculation Guidance

For KL divergence, use smoothed discrete distributions so no bin has zero probability:

```
P_smooth[i] = (P[i] + epsilon) / sum(P + epsilon)
Q_smooth[i] = (Q[i] + epsilon) / sum(Q + epsilon)
KL(P || Q) = sum(P_smooth[i] * log(P_smooth[i] / Q_smooth[i]))
```

For Jensen-Shannon divergence:

```
M = 0.5 * (P_smooth + Q_smooth)
JS(P, Q) = 0.5 * KL(P_smooth || M) + 0.5 * KL(Q_smooth || M)
```

Use behavior distributions that make sense for the scenario: open-port counts, service classes, response codes, dependency counts, or replay outcomes. If a scenario is a single critical rule change and the aggregate distribution barely moves, say so. That is a finding, not a failure.

Analysis Questions

1. **Threshold failure.** Which scenario is easiest for KL or JS divergence to miss? Why can a small distribution change still invalidate a remediation decision?

2. **Metric pairing.** For each scenario, identify the best companion metric: device-count delta, explicit firewall diff, dependency diff, replay mismatch rate, fallback-profile count, or freshness policy. Which companion metric should be mandatory before remediation approval?

3. **False positive / false negative analysis.** Identify one case where drift metrics could overreact and one case where they could underreact. What human review or corroborating evidence would correct each case?

4. **Authority update.** Revisit your Twin Authority Gate from Exercise 9. Does the final recommendation remain Trust, become Trust with limits, require Resync/Rebuild, or become Block? State the exact condition that changed the verdict.

5. **Research memo.** Write a short memo answering: "When should a digital twin lose authority to approve remediation?" Your answer must separate model fidelity, evidence freshness, operational materiality, and authorization.

Cleanup

```
# Delete the twin
curl -s -X DELETE "http://localhost:8100/v1/twin/$TWIN_ID" \
  -H "Authorization: Bearer $TOKEN" | jq .

# Stop the lab environment
cd student-lab/ && docker compose down
```

Troubleshooting Reference

Symptom	Cause	Fix
Twin deploy returns 404	Scan not found or not completed	Verify SCAN_ID and scan status
Scenario returns empty results	Steps target IPs not in topology	Check twin device IPs
Remediation action_failed > 0	Target IP not in twin	Verify device IPs with twin status
Cascading failures empty	No inter-device dependencies	Normal for simple topologies
Traffic replay returns 0 responses	Target port not in twin profile	Verify port is mapped in profile

---

Slide Reference Index

Exercise	Topic	Slides
1	Twin fidelity and profile mapping	001-015
2	Attack scenario design and determinism	016-025
3	Remediation strategy comparison	026-040
4	Cascading failure diagnosis	041-050
5	Constrained patch ordering	051-058
6	Zero-disruption validation	059-065
7	Traffic replay fidelity	066-072
8	Risk comparison dashboard	073-078
9	End-to-end remediation pipeline	079-085
10	Enterprise twin strategy	086-095
11	Twin authority under drift	096-105