SEAS-8414 Week 07 Student Lab Guide

SEAS-8414 Week 07 Student Lab Guide: Post-Quantum Cryptographic Readiness

Start Here: Download the Student ZIP

Before running any lab commands, download the Week 07 student package from the course site:

https://8414.bwater.io/downloads/labs/packages/seas8414-blackboard-week-07-2026.05.0_4e76a52f_aws8414.zip

The recommended path is the package Makefile:

```
unzip seas8414-blackboard-week-07-2026.05.0_4e76a52f_aws8414.zip
cd seas8414-blackboard-week-07-2026.05.0_4e76a52f_aws8414
make week07
```

The Makefile calls `run-week07-lab.sh`, extracts the nested runtime ZIP, starts the lab, runs the Week 07 API workflow, saves evidence under `lab-results/week-07/evidence/`, generates `lab-results/week-07/index.html`, and cleans up containers at exit.

You can also download the runner directly from https://8414.bwater.io/downloads/labs/scripts/run-week07-lab.sh.

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

```
unzip seas8414-blackboard-week-07-2026.05.0_4e76a52f_aws8414.zip
cd seas8414-blackboard-week-07-2026.05.0_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/phase07-lab-screencast.mp4
- LLM Prompt: https://8414.bwater.io/downloads/labs/prompts/phase07-lab-llm-prompt.md
- Run Script: https://8414.bwater.io/downloads/labs/scripts/run-week07-lab.sh

---

Breakwater Phase 7: Post-Quantum Cryptographic Readiness Lab

Phase 7 introduces post-quantum cryptographic readiness assessment: cipher suite scanning against NIST PQC standards, Harvest-Now-Decrypt-Later (HNDL) risk modelling, X.509 certificate lifecycle analysis, cryptographic agility testing, PQ migration planning with cost estimation, hybrid certificate chain validation, and HYDRA stream aggregation. These capabilities move Breakwater from identifying known CVEs to forecasting the quantum threat horizon and building a migration roadmap before cryptographically relevant quantum computers arrive.

(See Slides 001-005 for Phase 7 overview, quantum threat timeline, and NIST PQC standardisation)

Prerequisites

- Completed Phase 1 lab (scan data with TLS-enabled devices in the lab environment)
- Completed Phase 4 lab (attack graph and BRS scores computed)
- A completed scan with `$SCAN_ID` and `$TOKEN` variables set (see Phase 1 Exercise 6)
- Basic understanding of asymmetric cryptography (RSA, ECDHE, key sizes)
- Familiarity with TLS handshake concepts (cipher suite negotiation)

If you need to set up your variables from a previous session:

```
# Login and capture token
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')

# Get the most recent completed scan ID
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"
```

What you should see:

```
Token: eyJhbGciOiJIUzI1Ni...
Scan ID: a3f1c2d4-5e6f-7890-abcd-ef1234567890
```

> **Troubleshooting:** If `Token: null` appears, verify the user account exists. If `Scan ID: null`, no completed scan exists -- run `python scan_report.py 172.30.0.0/24` to create one.

Phase 7 API Cheatsheet

Endpoint	Method	Description
`/v1/quantum/{scan_id}`	GET	Quantum readiness scorecard (org-wide)
`/v1/quantum/{scan_id}/hndl`	GET	HNDL risk analysis per device
`/v1/quantum/{scan_id}/migration`	GET	PQ migration plan with priorities
`/v1/quantum/{scan_id}/certs`	GET	X.509 certificate lifecycle analysis
`/v1/quantum/{scan_id}/hydra`	GET	HYDRA stream 7 aggregated scorecard

All Phase 7 endpoints require Bearer token authentication.

(See Slides 081-086 for API design, endpoint reference, and database models)

Exercise 1: Get the Quantum Readiness Scorecard

(See Slides 010-022 for quantum threat background, NIST PQC standards, and scoring methodology)

The quantum readiness scorecard aggregates every device's cryptographic posture into a single organisation-wide score from 0 (fully vulnerable) to 100 (fully quantum-safe). It classifies devices as `vulnerable` (using classical algorithms that a CRQC can break), `transitional` (using extended key sizes that delay but do not eliminate the threat), or `safe` (using NIST-standardised post-quantum algorithms).

Architecture: What This Exercise Tests

```
graph LR
    subgraph "Crypto Scanner (crypto_scanner.py)"
        A[TLS Handshake<br/>from Enrichment] --> B[CipherAssessment]
        C[TLS Certificate<br/>from Phase 2] --> B
        B --> D[CryptoPosture per device]
```

```
    end
    subgraph "PQ Classifier (pq_classifier.py)"
        D --> E{Key Exchange?}
        E -->|ML-KEM, CRYSTALS| F[safe]
        E -->|ECDHE-4096+| G[transitional]
        E -->|RSA, ECDHE < 4096| H[vulnerable]
    end
    subgraph "Scoring (quantum_phase.py)"
        F & G & H --> I[QuantumReadinessScore]
        I --> J["GET /v1/quantum/{scan_id}"]
    end
```

Step 1: Retrieve the org-wide quantum readiness scorecard

```
# Full scorecard response
curl -s http://localhost:8100/v1/quantum/$SCAN_ID \
  -H "Authorization: Bearer $TOKEN" | jq .

# Summary metrics only
curl -s http://localhost:8100/v1/quantum/$SCAN_ID \
  -H "Authorization: Bearer $TOKEN" \
  | jq '{overall_score: .data.overall_score,
         device_count: .data.device_count,
         vulnerable_count: .data.vulnerable_count,
         transitional_count: .data.transitional_count,
         safe_count: .data.safe_count}'
```

What you should see:

```
{
  "overall_score": 12.5,
  "device_count": 20,
  "vulnerable_count": 17,
  "transitional_count": 2,
  "safe_count": 1
}
```

A score of 12.5 reflects a typical IoT environment: nearly all devices use legacy RSA-2048 or ECDHE-256 cipher suites that a cryptographically relevant quantum computer (CRQC) could break in hours.

Step 2: Examine top risks

```
# Top risk items requiring immediate attention
curl -s http://localhost:8100/v1/quantum/$SCAN_ID \
  -H "Authorization: Bearer $TOKEN" \
  | jq '[.data.top_risks[] | {ip: .device_ip, risk: .risk_description,
        quantum_score: .quantum_score, vulnerability_rating:
        .vulnerability_rating}]'
```

```
# Count devices by vulnerability rating
curl -s http://localhost:8100/v1/quantum/$SCAN_ID \
  -H "Authorization: Bearer $TOKEN" \
  | jq '{vulnerable: .data.vulnerable_count,
         transitional: .data.transitional_count,
         safe: .data.safe_count,
         total: .data.device_count,
         pct_vulnerable: (.data.vulnerable_count / .data.device_count
        * 100 | round)}'
```

What you should see for top risks:

```
[
  { "ip": "172.30.0.10", "risk": "RSA-2048 TLS key exchange, HNDL
        critical",
    "quantum_score": 5.0, "vulnerability_rating": "vulnerable" },
  { "ip": "172.30.0.30", "risk": "RSA-2048 TLS key exchange, HNDL
        critical",
    "quantum_score": 5.0, "vulnerability_rating": "vulnerable" },
  { "ip": "172.30.0.32", "risk": "ECDHE-256 TLS key exchange, HNDL
        warning",
    "quantum_score": 22.0, "vulnerability_rating": "transitional" }
]
```

Step 3: View HNDL and migration summaries

```
# HNDL risk summary from scorecard
curl -s http://localhost:8100/v1/quantum/$SCAN_ID \
  -H "Authorization: Bearer $TOKEN" \
  | jq '.data.hndl_summary'

# Migration plan summary from scorecard
curl -s http://localhost:8100/v1/quantum/$SCAN_ID \
  -H "Authorization: Bearer $TOKEN" \
  | jq '.data.migration_summary'
```

What you should see for HNDL summary:

```
{
  "critical_count": 12,
  "warning_count": 5,
  "safe_count": 3,
  "avg_risk_score": 71.4
}
```

> **Troubleshooting:** If `overall_score: null` or the endpoint returns 404, the scan may not have completed the Phase 7 quantum assessment stage. Verify with: `curl -s "http://localhost:8100/v1/scanning/smart-scan/history?limit=1" -H "Authorization: Bearer $TOKEN" | jq`

> `'.scans[0].status'` -- it should be `"completed"`. If the scan completed but no quantum data exists, TLS services may not have been discovered in your network segment.

Questions:

1. What is the overall quantum readiness score for your scan? How many devices are classified as `vulnerable`, `transitional`, and `safe`? What percentage of devices require quantum migration? *(See Slides 018-020 for the scoring formula and classification thresholds)*
2. The score is a composite of individual device quantum scores (0-100). A device using ML-KEM-768 key exchange would score close to 100. A device using RSA-2048 would score close to 0. What score would you expect for a device using ECDHE-P384 (a classical curve with a 192-bit security level)? Why is a larger classical key size considered `transitional` rather than `safe`? *(See Slide 021 for the classification algorithm)*
3. Look at the `top_risks` list. What is the most common vulnerability type? Are the highest-risk devices the same ones that had the highest BRS scores in Phase 4? Why might cryptographic risk be distributed differently from CVE-based risk? *(See Slide 022 for the relationship between quantum risk and traditional vulnerability scoring)*
4. The `migration_summary` shows the total estimated cost and timeline for org-wide PQ migration. What factors drive these estimates? If your organisation deferred migration for 3 years, how would the HNDL risk change? *(See Slides 014-016 for quantum timeline models and CRQC arrival probability distributions)*

Exercise 2: Identify Vulnerable Cipher Suites

(See Slides 023-035 for cipher suite taxonomy, quantum vulnerability ratings, and NIST PQC algorithm families)

The crypto scanner probes each device's TLS endpoint and assesses every negotiated cipher suite against NIST's post-quantum classification. Classical key exchange algorithms (RSA, ECDHE, DHE) are vulnerable because Shor's algorithm can solve the integer factorisation and discrete logarithm problems they depend on in polynomial time on a CRQC.

Architecture: What This Exercise Tests

```
graph LR
    subgraph "Cipher Catalog (cipher_catalog.py)"
        A[700+ known cipher suites] --> B[Lookup by IANA name]
        B --> C[CipherAssessment]
    end
    subgraph "TLS Probe (crypto_scanner.py)"
        D[TLS ClientHello] --> E[ServerHello]
        E --> C
    end
    subgraph "PQ Rating (pq_classifier.py)"
```

```
        C --> F{key_exchange algorithm}
        F -->|ML-KEM-768/1024| G["safe — NIST FIPS 203"]
        F -->|ECDHE-P521, DHE-8192| H["transitional — extended
classical"]
        F -->|RSA, ECDHE-P256, ECDHE-P384| I["vulnerable — Shor-
breakable"]
    end
```

Step 1: List all cipher suites across the scan

```
# All device cipher assessments
curl -s http://localhost:8100/v1/quantum/$SCAN_ID \
  -H "Authorization: Bearer $TOKEN" \
  | jq '[.data.top_risks[] | {ip: .device_ip, vulnerability_rating:
        .vulnerability_rating, quantum_score: .quantum_score}] |
        sort_by(.quantum_score)'

# Unique vulnerability ratings
curl -s http://localhost:8100/v1/quantum/$SCAN_ID \
  -H "Authorization: Bearer $TOKEN" \
  | jq '[.data.top_risks[].vulnerability_rating] | group_by(.) |
        map({rating: .[0], count: length})'
```

What you should see:

```
[
  { "ip": "172.30.0.10", "vulnerability_rating": "vulnerable",
        "quantum_score": 5.0 },
  { "ip": "172.30.0.11", "vulnerability_rating": "vulnerable",
        "quantum_score": 5.0 },
  { "ip": "172.30.0.20", "vulnerability_rating": "vulnerable",
        "quantum_score": 5.0 },
  { "ip": "172.30.0.30", "vulnerability_rating": "vulnerable",
        "quantum_score": 5.0 },
  { "ip": "172.30.0.32", "vulnerability_rating": "transitional",
        "quantum_score": 22.0 }
]
```

Step 2: Inspect per-device cipher details

```
# Get full HNDL + cipher detail for a specific device
DEVICE_IP="172.30.0.10"

curl -s "http://localhost:8100/v1/quantum/$SCAN_ID/hndl?ip=$DEVICE_IP"
        \
  -H "Authorization: Bearer $TOKEN" \
  | jq '.data.risks[0] | {device_ip, data_sensitivity, risk_score,
        risk_category, quantum_arrival_years}'

# Compare cipher suites: camera vs. NAS vs. router
for IP in 172.30.0.10 172.30.0.30 172.30.0.32; do
  echo "--- $IP ---"
```

```
  curl -s "http://localhost:8100/v1/quantum/$SCAN_ID/hndl?ip=$IP" \
    -H "Authorization: Bearer $TOKEN" \
    | jq '.data.risks[0] | {ip: .device_ip, risk_score, risk_category,
        data_sensitivity}'
done
```

What you should see for the camera:

```
{
  "device_ip": "172.30.0.10",
  "data_sensitivity": "high",
  "risk_score": 82.5,
  "risk_category": "critical",
  "quantum_arrival_years": 7.0
}
```

Step 3: Summarise cipher vulnerability distribution

```
# Get all HNDL risks to build cipher distribution picture
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/hndl \
  -H "Authorization: Bearer $TOKEN" \
  | jq '[.data.risks[] | .risk_category] | group_by(.) |
        map({category: .[0], count: length})'

# Average risk score across all devices
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/hndl \
  -H "Authorization: Bearer $TOKEN" \
  | jq '(.data.risks | map(.risk_score) | add / length | round) as
        $avg | {avg_hndl_risk: $avg, device_count: .data.count}'
```

What you should see:

```
[
  { "category": "critical", "count": 12 },
  { "category": "warning",  "count": 5  },
  { "category": "safe",     "count": 3  }
]

{ "avg_hndl_risk": 71, "device_count": 20 }
```

> **Troubleshooting:** If `hndl` returns `"count": 0`, the quantum phase may not have found any TLS services. IoT devices running plain HTTP on port 80 will not have cipher suite data. Check whether your scan includes devices with HTTPS (port 443), RTSP over TLS (port 322), or other TLS-wrapped services.

Questions:

1. How many devices use quantum-vulnerable cipher suites? What is the most common key exchange algorithm in your scan? Explain why ECDHE (Elliptic Curve Diffie-Hellman Ephemeral) is vulnerable to Shor's algorithm despite being considered "modern" cryptography by classical standards. *(See Slides 025-028 for Shor's algorithm and key exchange vulnerability analysis)*

2. The vulnerability ratings are: `safe` (ML-KEM/ML-DSA), `transitional` (extended classical), and `vulnerable` (standard classical). What specific key size would an RSA device need to be considered `transitional`? Why is this only a delay tactic rather than a long-term solution? *(See Slide 031 for transitional classification criteria)*
3. A camera (data sensitivity: `high`) and a sensor (data sensitivity: `low`) both use RSA-2048. They have the same cipher vulnerability but potentially different HNDL risk scores. Why? What data factors drive the difference? *(See Slide 035 for data sensitivity scoring)*
4. NIST finalised ML-KEM (FIPS 203), ML-DSA (FIPS 204), and SLH-DSA (FIPS 205) in 2024. For a firmware vendor supporting IoT cameras, what is the minimum change needed to their TLS stack to move from `vulnerable` to `safe` for key exchange? What additional change is needed for authentication? *(See Slides 032-034 for NIST PQC algorithm families and migration paths)*

Exercise 3: Analyse the HNDL Risk Profile

(See Slides 036-048 for HNDL threat model, storage economics, and Monte Carlo timeline simulation)

Harvest-Now-Decrypt-Later (HNDL) is the strategy where adversaries capture encrypted traffic today and store it until a CRQC becomes available to decrypt it. This threat is relevant right now -- traffic captured in 2026 could be decrypted in 2033 if CRQC estimates are accurate. The risk depends on: data sensitivity (how valuable is the plaintext?), data shelf life (how long does the data remain sensitive?), and the CRQC arrival timeline.

Architecture: What This Exercise Tests

```
graph LR
    subgraph "HNDL Engine (hndl_engine.py)"
        A[CryptoPosture<br/>per device] --> B[HarvestRisk]
        C[data_sensitivity<br/>classifier] --> B
        D[Quantum Timeline<br/>Model] --> B
        B --> E["risk_score = f(sensitivity, shelf_life, arrival)"]
    end
    subgraph "Timeline Model (quantum_timeline.py)"
        D --> F["P(CRQC by year) via<br/>Moore's Law + research
milestones"]
        F --> G[quantum_arrival_years: 7.0]
    end
    subgraph "API"
        E --> H["GET /v1/quantum/{scan_id}/hndl"]
    end
```

Step 1: View HNDL risks for all devices

```
# All HNDL risks sorted by risk score
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/hndl \
  -H "Authorization: Bearer $TOKEN" \
  | jq '[.data.risks[] | {ip: .device_ip, risk_score, risk_category,
         data_sensitivity, data_shelf_life_years}] |
         sort_by(-.risk_score)'

# Count by risk category
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/hndl \
  -H "Authorization: Bearer $TOKEN" \
  | jq '[.data.risks[].risk_category] | group_by(.) | map({category: .
         [0], count: length})'
```

What you should see for top 5 devices by HNDL risk:

```
[
  { "ip": "172.30.0.10", "risk_score": 82.5, "risk_category":
         "critical",
    "data_sensitivity": "high", "data_shelf_life_years": 15 },
  { "ip": "172.30.0.11", "risk_score": 82.5, "risk_category":
         "critical",
    "data_sensitivity": "high", "data_shelf_life_years": 15 },
  { "ip": "172.30.0.30", "risk_score": 78.0, "risk_category":
         "critical",
    "data_sensitivity": "high", "data_shelf_life_years": 10 },
  { "ip": "172.30.0.31", "risk_score": 75.0, "risk_category":
         "critical",
    "data_sensitivity": "high", "data_shelf_life_years": 10 },
  { "ip": "172.30.0.50", "risk_score": 45.0, "risk_category":
         "warning",
    "data_sensitivity": "medium", "data_shelf_life_years": 5 }
]
```

Step 2: Inspect timeline details for a critical-risk device

```
# Full HNDL risk detail including timeline model parameters
curl -s "http://localhost:8100/v1/quantum/$SCAN_ID/hndl?
         ip=172.30.0.10" \
  -H "Authorization: Bearer $TOKEN" \
  | jq '.data.risks[0] | {device_ip, risk_score, risk_category,
         data_sensitivity,
       data_shelf_life_years, quantum_arrival_years,
         storage_cost_per_gb, timeline_details}'
```

What you should see:

```
{
  "device_ip": "172.30.0.10",
  "risk_score": 82.5,
  "risk_category": "critical",
  "data_sensitivity": "high",
  "data_shelf_life_years": 15,
```

```
  "quantum_arrival_years": 7.0,
  "storage_cost_per_gb": 0.02,
  "timeline_details": {
    "harvest_window_years": 7.0,
    "data_still_sensitive_at_crqc": true,
    "economic_viability": "highly_profitable",
    "required_storage_tb": 0.12,
    "storage_cost_usd": 2.46
  }
}
```

The timeline details show that at current storage costs of $0.02/GB/year, storing 7 years of encrypted camera traffic costs only $2.46 -- making HNDL economically viable for even low-budget attackers.

Step 3: Compare camera vs. sensor HNDL profiles

```
# Compare high-sensitivity vs. low-sensitivity devices
for IP in 172.30.0.10 172.30.0.50; do
  echo "--- $IP ---"
  curl -s "http://localhost:8100/v1/quantum/$SCAN_ID/hndl?ip=$IP" \
    -H "Authorization: Bearer $TOKEN" \
    | jq '.data.risks[0] | {ip: .device_ip, risk_score, category:
        .risk_category,
         sensitivity: .data_sensitivity, shelf_life:
        .data_shelf_life_years}'
done
```

What you should see:

```
--- 172.30.0.10 ---
{ "ip": "172.30.0.10", "risk_score": 82.5, "category": "critical",
  "sensitivity": "high", "shelf_life": 15 }
--- 172.30.0.50 ---
{ "ip": "172.30.0.50", "risk_score": 45.0, "category": "warning",
  "sensitivity": "medium", "shelf_life": 5 }
```

> **Troubleshooting:** If `risk_score: 0.0` for all devices, the quantum timeline model may not have run. This happens when TLS probe data is unavailable. In the IoT simulation lab, devices use plaintext HTTP; HNDL risk is computed from device type inference rather than live TLS probing.

Questions:

1. What is the HNDL risk score for the highest-risk device? Which factors contribute most to its score: data sensitivity, data shelf life, or quantum arrival timeline? What would the risk score be if the same device were running ML-KEM (post-quantum key exchange)? *(See Slides 040-043 for the HNDL risk formula and worked examples)*
2. The `economic_viability` field is `"highly_profitable"` for cameras. The storage cost for 7 years of encrypted video is calculated as $2.46. Walk through the

calculation: `size_gb * years * cost_per_gb_per_year`. What assumptions does this calculation make, and how might real adversaries (nation-state vs. criminal) differ in their storage economics? *(See Slide 044 for the storage cost model)*

3. A device with `data_shelf_life_years: 5` and `quantum_arrival_years: 7.0` has a different risk profile than one with `data_shelf_life_years: 15`. Explain why: if the CRQC arrives in year 7 but the data is only sensitive for 5 years, what is the attacker's advantage window? *(See Slide 041 for the shelf-life vs. arrival overlap model)*
4. Which organisations face the highest HNDL risk today, and why? Consider: (a) national security agencies, (b) healthcare providers, (c) financial institutions, (d) smart home device makers. Rank them by HNDL urgency and justify your ranking. *(See Slides 045-048 for sector-specific HNDL threat analysis)*

Exercise 4: Inspect Certificate Lifecycle

(See Slides 049-058 for X.509 quantum vulnerability, certificate lifecycle tracking, and post-quantum certificates)

The certificate lifecycle analyser extracts X.509 certificates from TLS endpoints, checks their expiry dates, and assesses whether the underlying key algorithms (RSA, ECDSA) are vulnerable to Shor's algorithm. A certificate using RSA-2048 is quantum-vulnerable regardless of its expiry date -- but a certificate expiring in 2031 will need a post-quantum replacement anyway, making the timing an opportunity for migration.

Architecture: What This Exercise Tests

```
graph LR
    subgraph "Certificate Extraction (cert_lifecycle.py)"
        A[TLS Certificate<br/>from Phase 2] --> B[CertLifecycle]
        B --> C1[subject_cn / issuer_cn]
        B --> C2[not_after / days_remaining]
        B --> C3[key_type / key_size]
        B --> C4[quantum_vulnerable: bool]
        B --> C5[chain_issues list]
    end
    subgraph "Quantum Assessment"
        C3 --> D{Key algorithm?}
        D -->|RSA, ECDSA, ECDH| E[quantum_vulnerable: true]
        D -->|ML-DSA, SLH-DSA| F[quantum_vulnerable: false]
    end
    subgraph "API"
        B --> G["GET /v1/quantum/{scan_id}/certs"]
    end
```

Step 1: List all certificates in the scan

```
# All certificate lifecycle data
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/certs \
  -H "Authorization: Bearer $TOKEN" \
  | jq '{count: .data.count, certs: [.data.certs[] |
        {ip: .device_ip, subject: .subject_cn, expires: .not_after,
         days_remaining, key_type, key_size, quantum_vulnerable}]}'

# Separate quantum-vulnerable from safe
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/certs \
  -H "Authorization: Bearer $TOKEN" \
  | jq '{vulnerable: [.data.certs[] | select(.quantum_vulnerable ==
        true) | .device_ip],
         safe: [.data.certs[] | select(.quantum_vulnerable == false) |
        .device_ip]}'
```

What you should see:

```
{
  "count": 8,
  "certs": [
    { "ip": "172.30.0.10", "subject": "172.30.0.10",
      "expires": "2027-06-15T00:00:00Z", "days_remaining": 467,
      "key_type": "RSA", "key_size": 2048, "quantum_vulnerable": true
        },
    { "ip": "172.30.0.30", "subject": "QNAP NAS",
      "expires": "2026-09-01T00:00:00Z", "days_remaining": 180,
      "key_type": "RSA", "key_size": 2048, "quantum_vulnerable": true
        },
    { "ip": "172.30.0.32", "subject": "Router Management",
      "expires": "2028-01-20T00:00:00Z", "days_remaining": 686,
      "key_type": "EC", "key_size": 256, "quantum_vulnerable": true }
  ]
}
```

Step 2: Check for chain issues and expiring certificates

```
# Certificates expiring within 365 days
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/certs \
  -H "Authorization: Bearer $TOKEN" \
  | jq '[.data.certs[] | select(.days_remaining < 365) |
        {ip: .device_ip, subject: .subject_cn, days_remaining,
        key_type, chain_issues}]'

# Certificates with chain issues
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/certs \
  -H "Authorization: Bearer $TOKEN" \
  | jq '[.data.certs[] | select(.chain_issues | length > 0) |
        {ip: .device_ip, issues: .chain_issues}]'
```

What you should see for expiring certificates:

```
[
  {
    "ip": "172.30.0.30",
    "subject": "QNAP NAS",
    "days_remaining": 180,
    "key_type": "RSA",
    "chain_issues": ["Self-signed certificate -- no chain validation",
        "RSA-2048 CA key is quantum-vulnerable"]
  }
]
```

Step 3: Inspect a specific device's full certificate details

```
# Full cert detail for the NAS (expiring soonest)
curl -s "http://localhost:8100/v1/quantum/$SCAN_ID/certs?
         ip=172.30.0.30" \
  -H "Authorization: Bearer $TOKEN" \
  | jq '.data.certs[0]'

# Compare RSA-2048 vs. EC-256 certificates
for IP in 172.30.0.10 172.30.0.32; do
  echo "--- $IP ---"
  curl -s "http://localhost:8100/v1/quantum/$SCAN_ID/certs?ip=$IP" \
    -H "Authorization: Bearer $TOKEN" \
    | jq '.data.certs[0] | {ip: .device_ip, key_type, key_size,
         quantum_vulnerable, days_remaining}'
done
```

What you should see:

```
--- 172.30.0.10 ---
{ "ip": "172.30.0.10", "key_type": "RSA", "key_size": 2048,
  "quantum_vulnerable": true, "days_remaining": 467 }
--- 172.30.0.32 ---
{ "ip": "172.30.0.32", "key_type": "EC", "key_size": 256,
  "quantum_vulnerable": true, "days_remaining": 686 }
```

Both RSA-2048 and EC-256 are quantum-vulnerable despite EC being considered stronger for classical adversaries. This is because Shor's algorithm also solves the elliptic curve discrete logarithm problem (ECDLP).

> **Troubleshooting:** If `count: 0`, the scan did not discover any HTTPS or TLS-wrapped services. In the IoT simulation network, most devices use plaintext HTTP. Certificate data is populated from Phase 2 enrichment (TLS inspection). Check: `curl -s http://localhost:8100/v1/analytics/graph/$SCAN_ID -H "Authorization: Bearer $TOKEN" | jq '[.data.nodes[] | select(.metadata.port == 443)] | length'` to verify HTTPS services were found.

Questions:

1. How many unique certificates were discovered? What percentage use RSA keys vs. EC keys? Both are quantum-vulnerable -- but which is more commonly deployed in IoT devices, and why? *(See Slide 052 for RSA vs. EC deployment statistics in embedded systems)*
2. The NAS certificate expires in 180 days. The organisation must renew it anyway. Explain why this renewal is the ideal opportunity to migrate to a post-quantum certificate (ML-DSA). What would the migration process look like: CSR generation, CA requirements, device firmware support? *(See Slides 055-057 for the certificate renewal migration workflow)*
3. Examine the `chain_issues` list. One issue is "Self-signed certificate." What security implication does self-signing have for a production IoT device? Why do most embedded systems ship with self-signed certificates, and what are the alternatives? *(See Slide 054 for certificate chain analysis)*
4. A post-quantum certificate using ML-DSA has a public key size of approximately 1312 bytes -- compared to 294 bytes for a P-256 EC key. What are the practical implications of larger post-quantum keys for IoT devices with constrained memory and bandwidth? *(See Slides 058 for PQ certificate size comparison and IoT constraints)*

Exercise 5: Test Cryptographic Agility

(See Slides 059-068 for cryptographic agility definition, hybrid handshake design, and agility scoring)

Cryptographic agility is a device's ability to negotiate post-quantum cipher suites. An agile device can perform a hybrid TLS handshake combining a classical algorithm (for current interoperability) with a post-quantum algorithm (for forward security). The agility scanner tests whether each device accepts ML-KEM-768 or X25519Kyber768 key exchange.

Architecture: What This Exercise Tests

```
graph LR
    subgraph "Agility Scanner (agility_scanner.py)"
        A[Device TLS Endpoint] --> B[PQ ClientHello<br/>with ML-KEM-
768]
        B --> C{Server Response}
        C -->|Accepts PQ KEM| D[agile: true]
        C -->|Rejects / falls back| E[agile: false]
    end
    subgraph "PQ Validator (pq_validator.py)"
        D --> F[Hybrid cert chain check<br/>ML-DSA + RSA signature]
        F --> G[PQValidationResult]
    end
    subgraph "API"
        G --> H["GET /v1/quantum/{scan_id}/migration<br/>(includes
```

```
agility assessment)"]
    end
```

Step 1: Check agility data via the migration plan

```
# Migration plan includes agility assessment per device
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/migration \
  -H "Authorization: Bearer $TOKEN" \
  | jq '{scan_id: .data.scan_id,
         total_devices: .data.total_devices,
         vulnerable_devices: .data.vulnerable_devices,
         risk_reduction_pct: .data.risk_reduction_pct}'

# List migration actions with action types (agility is reflected in
        action type)
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/migration \
  -H "Authorization: Bearer $TOKEN" \
  | jq '[.data.actions[] | {ip: .device_ip, action_type, priority,
        description, estimated_cost}] | sort_by(.priority)'
```

What you should see for migration summary:

```
{
  "scan_id": "a3f1c2d4-...",
  "total_devices": 20,
  "vulnerable_devices": 17,
  "risk_reduction_pct": 85.0
}
```

What you should see for migration actions (top 5 by priority):

```
[
  { "ip": "172.30.0.10", "action_type": "vpn_overlay",
    "priority": 1, "description": "Deploy PQ-VPN overlay (ML-KEM-768)
        for camera",
    "estimated_cost": 250.0 },
  { "ip": "172.30.0.11", "action_type": "vpn_overlay",
    "priority": 1, "description": "Deploy PQ-VPN overlay (ML-KEM-768)
        for camera",
    "estimated_cost": 250.0 },
  { "ip": "172.30.0.30", "action_type": "firmware_update",
    "priority": 2, "description": "Firmware update with PQ TLS support
        (ML-KEM-768)",
    "estimated_cost": 1200.0 },
  { "ip": "172.30.0.32", "action_type": "gateway_proxy",
    "priority": 3, "description": "PQ-capable TLS termination proxy
        for router",
    "estimated_cost": 500.0 }
]
```

Step 2: Understand the four migration action types

```
# Count action types across the migration plan
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/migration \
  -H "Authorization: Bearer $TOKEN" \
  | jq '[.data.actions[].action_type] | group_by(.) | map({type: .[0],
        count: length})'

# Total cost and hours by action type
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/migration \
  -H "Authorization: Bearer $TOKEN" \
  | jq '[.data.actions | group_by(.action_type)[] |
       {type: .[0].action_type,
        count: length,
        total_cost: (map(.estimated_cost) | add),
        total_hours: (map(.estimated_hours) | add)}]'
```

What you should see:

```
[
  { "type": "firmware_update",  "count": 5,  "total_cost": 6000.0,
       "total_hours": 15.0 },
  { "type": "gateway_proxy",    "count": 3,  "total_cost": 1500.0,
       "total_hours": 9.0  },
  { "type": "vpn_overlay",      "count": 8,  "total_cost": 2000.0,
       "total_hours": 8.0  },
  { "type": "cert_rotation",    "count": 1,  "total_cost": 200.0,
       "total_hours": 2.0  }
]
```

> **Troubleshooting:** If migration plan returns 404, ensure the quantum assessment completed. The migration planner runs after cipher scanning. Check `curl -s http://localhost:8100/v1/quantum/$SCAN_ID -H "Authorization: Bearer $TOKEN" | jq '.data.device_count'` -- if zero, re-trigger the scan.

Questions:

1. The agility scanner assigns `vpn_overlay` as the migration action for devices that cannot be firmware-updated. Explain what a PQ-VPN overlay does: how does it provide post-quantum security without modifying the device firmware? What are the operational trade-offs? *(See Slides 062-064 for PQ-VPN overlay architecture)*
2. `firmware_update` is assigned to devices where the vendor provides a PQ-capable firmware version. What three conditions must be true for a firmware update to be the right migration action? (Consider: vendor PQ support, device age, update mechanism availability.) *(See Slide 065 for firmware migration decision tree)*
3. `gateway_proxy` terminates TLS at a PQ-capable gateway and re-encrypts to the legacy device. Draw a traffic flow diagram showing how data moves through a gateway proxy: Internet -> PQ Gateway -> Legacy Device. At what point in this path is the data quantum-safe, and at what point is it not? *(See Slide 066 for gateway proxy architecture)*
4. The migration plan projects 85% risk reduction if all actions are implemented. Why is this not 100%? What residual quantum risk remains after migrating all TLS cipher suites? Consider: data already harvested, non-TLS protocols, and cryptographic

libraries embedded in application code. *(See Slide 068 for residual risk after PQ migration)*

Exercise 6: Generate a Migration Plan

(See Slides 069-075 for migration planner algorithm, cost model, and timeline estimation)

The migration planner builds a prioritised action list for every vulnerable device, estimates the cost (licence fees, engineering hours, downtime) for each action, and projects the overall migration timeline. Actions are prioritised by HNDL risk score, then by action difficulty (firmware updates are harder than VPN overlays).

Architecture: What This Exercise Tests

```
graph LR
    subgraph "Migration Planner (migration_planner.py)"
        A[HarvestRisk per device] --> B[sort by risk_score desc]
        B --> C{Device agile?}
        C -->|No firmware support| D[vpn_overlay action]
        C -->|Vendor PQ firmware available| E[firmware_update action]
        C -->|Cannot update but routable| F[gateway_proxy action]
        C -->|Cert expiring < 1 year| G[cert_rotation action]
    end
    subgraph "Cost Estimator (cost_estimator.py)"
        D & E & F & G --> H[MigrationAction.estimated_cost]
        D & E & F & G --> I[MigrationAction.estimated_hours]
    end
    subgraph "API"
        H & I --> J["GET /v1/quantum/{scan_id}/migration"]
    end
```

Step 1: View the full migration plan

```
# Full migration plan
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/migration \
  -H "Authorization: Bearer $TOKEN" | jq .

# Plan totals
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/migration \
  -H "Authorization: Bearer $TOKEN" \
  | jq '{total_cost: .data.total_cost,
         timeline_months: .data.timeline_months,
         risk_reduction_pct: .data.risk_reduction_pct,
         action_count: (.data.actions | length)}'
```

What you should see:

```
{
  "total_cost": 9700.0,
  "timeline_months": 6,
  "risk_reduction_pct": 85.0,
  "action_count": 17
}
```

Step 2: Analyse cost breakdown by action type

```
# Cost per action sorted by cost descending
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/migration \
  -H "Authorization: Bearer $TOKEN" \
  | jq '[.data.actions[] | {ip: .device_ip, action_type, priority,
         estimated_cost, estimated_hours, description}]
        | sort_by(-.estimated_cost) | .[:5]'
```

What you should see:

```
[
  { "ip": "172.30.0.30", "action_type": "firmware_update", "priority":
        2,
    "estimated_cost": 1200.0, "estimated_hours": 4.0,
    "description": "Firmware update with PQ TLS support (ML-KEM-768)"
        },
  { "ip": "172.30.0.31", "action_type": "firmware_update", "priority":
        2,
    "estimated_cost": 1200.0, "estimated_hours": 4.0,
    "description": "Firmware update with PQ TLS support (ML-KEM-768)"
        },
  { "ip": "172.30.0.32", "action_type": "gateway_proxy", "priority":
        3,
    "estimated_cost": 500.0, "estimated_hours": 3.0,
    "description": "PQ-capable TLS termination proxy for router" }
]
```

Step 3: Build a phased rollout schedule

```
# Group actions by priority for phased rollout planning
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/migration \
  -H "Authorization: Bearer $TOKEN" \
  | jq '[.data.actions | group_by(.priority)[] |
        {phase: .[0].priority,
         action_count: length,
         phase_cost: (map(.estimated_cost) | add),
         phase_hours: (map(.estimated_hours) | add),
         ips: [.[].device_ip]}]'
```

What you should see:

```
[
  { "phase": 1, "action_count": 8, "phase_cost": 2000.0,
       "phase_hours": 8.0,
    "ips": ["172.30.0.10", "172.30.0.11", "172.30.0.12",
       "172.30.0.13",
          "172.30.0.20", "172.30.0.21", "172.30.0.40",
       "172.30.0.41"] },
  { "phase": 2, "action_count": 7, "phase_cost": 5800.0,
       "phase_hours": 19.0,
    "ips": ["172.30.0.30", "172.30.0.31", "172.30.0.42",
       "172.30.0.50",
          "172.30.0.51", "172.30.0.52", "172.30.0.60"] },
  { "phase": 3, "action_count": 2, "phase_cost": 1700.0,
       "phase_hours": 5.0,
    "ips": ["172.30.0.32", "172.30.0.70"] }
]
```


> **Troubleshooting:** If `total_cost: 0.0` or the actions list is empty, the migration planner may not have run. This happens when `vulnerable_devices: 0`. Check your quantum scorecard: `curl -s http://localhost:8100/v1/quantum/$SCAN_ID -H "Authorization: Bearer $TOKEN" | jq '.data.vulnerable_count'` -- if zero, all devices scored as `safe` or `transitional`.

Questions:

1. The plan estimates $9,700 total cost over 6 months for 17 actions. If an organisation has a quarterly security budget of $5,000, how would you phase the rollout to fit within budget? Identify which Phase 1 actions provide the highest risk reduction per dollar. *(See Slide 071 for cost optimisation in migration planning)*
2. `vpn_overlay` actions cost $250 per device while `firmware_update` actions cost $1,200. List three factors that make firmware updates more expensive despite being the preferred long-term solution. *(See Slide 073 for the cost model inputs)*
3. The migration timeline is 6 months. In a production IoT environment with 500+ devices, what organisational challenges would extend this timeline beyond the technical estimate? Consider: device access, maintenance windows, vendor coordination, testing, and staff training. *(See Slide 074 for enterprise migration timeline factors)*
4. The plan leaves 15% residual risk after implementation. Design a complementary control: a network-layer control (not cryptographic) that could reduce HNDL risk by intercepting or disrupting harvest-stage traffic. *(See Slide 075 for defence-in-depth for HNDL)*

Exercise 7: Estimate Migration Costs

(See Slides 076-080 for TCO breakdown: firmware, gateway, VPN, labour, and risk-adjusted ROI)

The cost estimator breaks down total cost of ownership (TCO) for the migration across three categories: firmware updates (vendor licensing, engineering time, device downtime), gateway/proxy deployment (hardware, software licensing, ongoing maintenance), and VPN overlay (client software, infrastructure, operational overhead).

Architecture: What This Exercise Tests

```
graph LR
    subgraph "Cost Estimator (cost_estimator.py)"
        A[action_type] --> B{Cost model}
        B -->|firmware_update| C["$200–1500 per device<br/>(size +
complexity)"]
        B -->|gateway_proxy| D["$500–2000 per zone<br/>(hardware +
setup)"]
        B -->|vpn_overlay| E["$150–350 per device<br/>(licence +
config)"]
        B -->|cert_rotation| F["$50–300 per device<br/>(CA + ops
time)"]
    end
    subgraph "TCO Output"
        C & D & E & F --> G[total_cost]
        C & D & E & F --> H[total_hours]
        G & H --> I[MigrationPlan.total_cost]
    end
```

Step 1: TCO breakdown by migration category

```
# Separate costs: firmware vs. gateway vs. VPN vs. cert
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/migration \
  -H "Authorization: Bearer $TOKEN" \
  | jq '
    .data.actions | group_by(.action_type) | map({
      category: .[0].action_type,
      devices: length,
      subtotal_cost: (map(.estimated_cost) | add),
      subtotal_hours: (map(.estimated_hours) | add),
      avg_cost_per_device: (map(.estimated_cost) | add / length |
        round)
    })
  '


# Grand totals
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/migration \
  -H "Authorization: Bearer $TOKEN" \
  | jq '{total_cost_usd: .data.total_cost,
         timeline_months: .data.timeline_months,
         risk_reduction_pct: .data.risk_reduction_pct,
         cost_per_point_risk_reduction: (.data.total_cost /
        .data.risk_reduction_pct | round)}'
```

What you should see for TCO breakdown:

```
[
  { "category": "cert_rotation",   "devices": 1, "subtotal_cost":
        200.0,   "subtotal_hours": 2.0,  "avg_cost_per_device": 200 },
  { "category": "firmware_update", "devices": 5, "subtotal_cost":
        6000.0,  "subtotal_hours": 15.0, "avg_cost_per_device": 1200
        },
  { "category": "gateway_proxy",   "devices": 3, "subtotal_cost":
        1500.0,  "subtotal_hours": 9.0,  "avg_cost_per_device": 500 },
  { "category": "vpn_overlay",     "devices": 8, "subtotal_cost":
        2000.0,  "subtotal_hours": 8.0,  "avg_cost_per_device": 250 }
]
```

Step 2: Risk-adjusted ROI calculation

```
# Calculate ROI: cost of migration vs. cost of breach
# Assume: median IoT breach cost = $4.5M (IBM Cost of Data Breach
        2024)
MIGRATION_COST=$(curl -s
        http://localhost:8100/v1/quantum/$SCAN_ID/migration \
  -H "Authorization: Bearer $TOKEN" \
  | jq '.data.total_cost')

RISK_REDUCTION=$(curl -s
        http://localhost:8100/v1/quantum/$SCAN_ID/migration \
  -H "Authorization: Bearer $TOKEN" \
  | jq '.data.risk_reduction_pct')

echo "Migration cost: \$$MIGRATION_COST"
echo "Risk reduction: $RISK_REDUCTION%"

# Manual ROI: (breach_cost * risk_reduction_pct / 100) -
        migration_cost
# = ($4,500,000 * 0.85) - $9,700 = $3,815,300 net benefit
echo "Expected breach cost avoided: \$$(echo "$RISK_REDUCTION * 45000"
        | bc) (assuming 1% breach probability)"
```

Step 3: Compare migration costs across device categories

```
# Cameras: mostly vpn_overlay (firmware can't be updated)
CAMERA_IPS="172.30.0.10 172.30.0.11 172.30.0.12"

# NAS: firmware_update (QNAP/Synology support PQ TLS)
NAS_IPS="172.30.0.30 172.30.0.31"

for IP in $CAMERA_IPS; do
  curl -s http://localhost:8100/v1/quantum/$SCAN_ID/migration \
    -H "Authorization: Bearer $TOKEN" \
    | jq --arg ip "$IP" '[.data.actions[] | select(.device_ip == $ip)
      | {ip: .device_ip, action_type, estimated_cost}]'
done
```

> **Troubleshooting:** If migration cost is unexpectedly zero for many devices, check whether they received `vulnerability_rating: "safe"`. Devices with no TLS services may default to safe. Use `--arg` to filter specific devices in jq queries.

Questions:

1. Which migration category (firmware, gateway, VPN, cert) has the highest cost per device? Which has the lowest? What explains the 5x cost difference between firmware updates and VPN overlays? *(See Slide 077 for cost model inputs and assumptions)*
2. The cost model estimates $250/device for a VPN overlay. List the specific components that make up this cost: software licence, configuration labour, testing, and ongoing operational overhead. How would this cost change for a fleet of 5,000 IoT cameras? *(See Slide 078 for VPN overlay cost breakdown)*
3. Calculate the risk-adjusted ROI for the migration: if the median post-quantum breach cost is $4.5M and the migration achieves 85% risk reduction, what is the net expected value of the investment? Use the formula: `EV = (breach_probability * breach_cost * risk_reduction) - migration_cost`. *(See Slide 079 for ROI calculation methodology)*
4. The cost estimator does not account for productivity loss during device downtime. For a 20-camera surveillance network where downtime means a security gap, estimate the indirect cost of firmware updates (4 hours downtime per device). How does this change the ROI calculation? *(See Slide 080 for indirect cost modelling)*

Exercise 8: Run PQ Validation

(See Slides 081-088 for hybrid certificate chain validation, ML-DSA signature verification, and compliance testing)

The PQ validator checks whether devices support hybrid certificate chains -- certificates that carry both a classical signature (RSA or ECDSA, for backward compatibility) and a post-quantum signature (ML-DSA, for forward quantum security). A passing validation means the device can participate in a hybrid TLS handshake during the migration transition period.

Architecture: What This Exercise Tests

```
graph LR
    subgraph "PQ Validator (pq_validator.py)"
        A[Device TLS Endpoint] --> B[PQ Test ClientHello<br/>hybrid
cipher suite]
        B --> C{Handshake result}
        C -->|Success| D[Check cert chain]
        D --> E{ML-DSA sig present?}
        E -->|Yes| F[pq_valid: true]
        E -->|No| G[pq_valid: false]
        C -->|Failure / fallback| H[pq_valid: false]
    end
```

```
    subgraph "FIPS 204 Compliance"
        F --> I[Verify ML-DSA params:<br/>ML-DSA-44 / 65 / 87]
        I --> J[nist_category: 2/3/5]
    end
    subgraph "API"
        F & G & H --> K["GET /v1/quantum/{scan_id}/migration<br/>
(pq_validation embedded)"]
    end
```

Step 1: View PQ validation results from migration plan

```
# Migration plan includes action types which reflect validation
        results
# Devices receiving "firmware_update" actions have been assessed as
        updatable
# Devices receiving "vpn_overlay" failed PQ validation (can't do
        native PQ TLS)

curl -s http://localhost:8100/v1/quantum/$SCAN_ID/migration \
  -H "Authorization: Bearer $TOKEN" \
  | jq '{
    firmware_update_count: [.data.actions[] | select(.action_type ==
        "firmware_update")] | length,
    vpn_overlay_count: [.data.actions[] | select(.action_type ==
        "vpn_overlay")] | length,
    gateway_proxy_count: [.data.actions[] | select(.action_type ==
        "gateway_proxy")] | length,
    cert_rotation_count: [.data.actions[] | select(.action_type ==
        "cert_rotation")] | length
  }'

# Devices that CAN do PQ natively (firmware_update = device supports
        PQ TLS natively)
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/migration \
  -H "Authorization: Bearer $TOKEN" \
  | jq '[.data.actions[] | select(.action_type == "firmware_update" or
        .action_type == "cert_rotation")
        | {ip: .device_ip, action_type, priority, description}]'
```

What you should see:

```
{
  "firmware_update_count": 5,
  "vpn_overlay_count": 8,
  "gateway_proxy_count": 3,
  "cert_rotation_count": 1
}
```

Step 2: Check HYDRA stream for PQ validation summary

```
# HYDRA provides the aggregated PQ posture including validation
        pass/fail
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/hydra \
  -H "Authorization: Bearer $TOKEN" \
  | jq '{overall_score: .data.scorecard.overall_score,
         exposure_pct: .data.exposure_pct,
         device_count: .data.device_count,
         hndl_heatmap: .data.hndl_heatmap}'
```

What you should see:

```
{
  "overall_score": 12.5,
  "exposure_pct": 85.0,
  "device_count": 20,
  "hndl_heatmap": {
    "critical": { "low": 0, "medium": 2, "high": 8, "critical": 2 },
    "warning":  { "low": 1, "medium": 3, "high": 1, "critical": 0 },
    "safe":     { "low": 2, "medium": 1, "high": 0, "critical": 0 }
  }
}
```

Step 3: Interpret the HNDL heatmap

The heatmap cross-references risk category (safe / warning / critical) with data sensitivity (low / medium / high / critical). The cell `hndl_heatmap.critical.high = 8` means 8 devices have CRITICAL HNDL risk AND handle HIGH-sensitivity data -- the top-priority migration targets.

```
# Find the intersection: critical HNDL risk + high data sensitivity
# These are the devices in the hndl_heatmap.critical.high cell
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/hndl \
  -H "Authorization: Bearer $TOKEN" \
  | jq '[.data.risks[] | select(.risk_category == "critical" and
        .data_sensitivity == "high")
        | {ip: .device_ip, risk_score, data_sensitivity}] |
        sort_by(-.risk_score)'
```

What you should see:

```
[
  { "ip": "172.30.0.10", "risk_score": 82.5, "data_sensitivity":
        "high" },
  { "ip": "172.30.0.11", "risk_score": 82.5, "data_sensitivity":
        "high" },
  { "ip": "172.30.0.12", "risk_score": 80.0, "data_sensitivity":
        "high" },
  { "ip": "172.30.0.13", "risk_score": 80.0, "data_sensitivity":
        "high" }
]
```

> **Troubleshooting:** If `exposure_pct: 0.0` in the HYDRA response, check that `get_quantum_assessments` returned data: `curl -s http://localhost:8100/v1/quantum/$SCAN_ID -H "Authorization: Bearer $TOKEN" | jq '.data.device_count'`. Zero device count means the quantum phase did not persist assessment results.

Questions:

1. How many devices passed PQ validation (received `firmware_update` action, meaning their firmware can be updated to support PQ TLS natively)? How many failed (received `vpn_overlay` or `gateway_proxy`)? What is the implication of a device failing PQ validation? *(See Slides 081-083 for PQ validation pass/fail criteria)*
2. A hybrid certificate carries two signatures: an RSA signature (valid for current browsers) and an ML-DSA signature (valid for PQ-capable clients). During a TLS 1.3 handshake, how does the client determine which signature to verify? What happens if the client is PQ-capable but the server's certificate only carries an RSA signature? *(See Slides 084-086 for hybrid handshake protocol detail)*
3. ML-DSA-44 provides NIST Category 2 security (equivalent to AES-128 against quantum attacks), while ML-DSA-87 provides Category 5 (equivalent to AES-256). For a 10-year certificate validity period and a CRQC estimated in 7 years, which ML-DSA security category should you select? Justify your choice. *(See Slide 087 for NIST category selection guidance)*
4. The PQ validator tests whether a device accepts a PQ ClientHello without crashing or hanging. Why might a firmware-level bug cause a device to crash on receiving an ML-KEM key share rather than simply falling back to a classical cipher? What testing should vendors perform to prevent this? *(See Slide 088 for PQ compatibility testing requirements)*

Exercise 9: View the HYDRA Stream

(See Slides 089-096 for HYDRA stream architecture, data sensitivity heatmap, and top-10 priority list)

HYDRA (Hybrid Data Risk Assessment) is the org-wide aggregation layer that combines all quantum readiness data into a single dashboard view. It produces a heatmap crossing HNDL risk category (safe / warning / critical) with data sensitivity (low / medium / high / critical), and identifies the top-10 priority devices for immediate migration action.

Architecture: What This Exercise Tests

```
graph LR
    subgraph "HYDRA Engine (hydra_crypto.py)"
        A[QuantumReadinessScore] --> B[HYDRA Builder]
        C[HarvestRisk list] --> B
        D[CryptoPosture list] --> B
    end
    subgraph "Heatmap Construction"
```

```
        B --> E[Cross-tab:<br/>risk_category x data_sensitivity]
        E --> F[4x3 heatmap matrix]
    end
    subgraph "Top-10 Priority"
        B --> G[Sort by risk_score desc]
        G --> H[Top-10 device list]
    end
    subgraph "API"
        F & H --> I["GET /v1/quantum/{scan_id}/hydra"]
    end
```

Step 1: Get the full HYDRA scorecard

```
# Full HYDRA response
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/hydra \
  -H "Authorization: Bearer $TOKEN" | jq .

# Key metrics
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/hydra \
  -H "Authorization: Bearer $TOKEN" \
  | jq '{overall_score: .data.scorecard.overall_score,
         exposure_pct: .data.exposure_pct,
         device_count: .data.device_count}'
```

What you should see:

```
{
  "overall_score": 12.5,
  "exposure_pct": 85.0,
  "device_count": 20
}
```

Step 2: Interpret the HNDL heatmap

```
# Full heatmap with labels
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/hydra \
  -H "Authorization: Bearer $TOKEN" \
  | jq '.data.hndl_heatmap'

# Find the highest-priority cell (critical risk + critical
        sensitivity)
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/hydra \
  -H "Authorization: Bearer $TOKEN" \
  | jq '.data.hndl_heatmap.critical.critical'
```

What you should see for the full heatmap:

```
{
  "critical": { "low": 0, "medium": 2, "high": 8, "critical": 2 },
  "warning":  { "low": 1, "medium": 3, "high": 1, "critical": 0 },
```

```
  "safe":     { "low": 2, "medium": 1, "high": 0, "critical": 0 }
}
```

Read the heatmap: the cell `critical.high = 8` means 8 devices have HNDL risk = critical AND handle high-sensitivity data. The cell `critical.critical = 2` is the most urgent category -- critical risk AND critical data (likely PLC/OT devices).

Step 3: View top risks from the HYDRA scorecard

```
# Top risks from scorecard embedded in HYDRA
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/hydra \
  -H "Authorization: Bearer $TOKEN" \
  | jq '[.data.scorecard.top_risks[] | {ip: .device_ip, quantum_score,
         vulnerability_rating, risk: .risk_description}] | .[:10]'

# Compare vulnerability counts
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/hydra \
  -H "Authorization: Bearer $TOKEN" \
  | jq '{vulnerable: .data.scorecard.vulnerable_count,
         transitional: .data.scorecard.transitional_count,
         safe: .data.scorecard.safe_count}'
```

What you should see for top risks:

```
[
  { "ip": "172.30.0.10", "quantum_score": 5.0, "vulnerability_rating":
       "vulnerable",
    "risk": "RSA-2048 TLS key exchange, HNDL critical" },
  { "ip": "172.30.0.11", "quantum_score": 5.0, "vulnerability_rating":
       "vulnerable",
    "risk": "RSA-2048 TLS key exchange, HNDL critical" },
  { "ip": "172.30.0.40", "quantum_score": 5.0, "vulnerability_rating":
       "vulnerable",
    "risk": "RSA-2048 TLS key exchange, HNDL critical" }
]
```

> **Troubleshooting:** If `hndl_heatmap` shows all zeros, the HNDL risk engine returned no results. This can happen when device data sensitivity classification defaults to `"unknown"`. In the IoT simulation, sensitivity is inferred from device type (camera = high, sensor = medium, PLC = critical). Check `curl -s http://localhost:8100/v1/quantum/$SCAN_ID/hndl -H "Authorization: Bearer $TOKEN" | jq '.data.count'`.

Questions:

1. The HYDRA heatmap has 12 cells (3 risk categories x 4 sensitivity levels). Which cell contains the most urgent devices? For the devices in that cell, what single migration action would provide the fastest risk reduction? *(See Slide 092 for heatmap quadrant analysis and remediation prioritisation)*
2. The HYDRA scorecard shows `exposure_pct: 85.0` (85% of devices are exposed to quantum risk). How does this metric differ from `vulnerable_count /`

`device_count`? What types of devices might be exposed but not rated `vulnerable`? *(See Slide 091 for exposure percentage calculation)*

3. The `top_risks` list is sorted by quantum score ascending (lowest score = highest risk, since 0 = fully vulnerable). Explain why the scoring is inverted from BRS (where higher = more risk). Design a normalisation that would make both scores consistent in direction. *(See Slide 090 for HYDRA score normalisation)*
4. A CISO wants to present the HYDRA heatmap to the board of directors. Translate the heatmap data into a business language summary: "X% of our devices are actively harvesting data that quantum computers will decrypt within Y years." Fill in the numbers and explain what action the board should authorise. *(See Slides 094-096 for board-level quantum risk communication)*

Exercise 10: Simulate the HNDL Timeline

(See Slides 097-105 for Monte Carlo HNDL simulation, threat economics model, and scenario comparison)

The threat simulator runs Monte Carlo simulations of the HNDL economics: projecting storage costs, quantum compute costs, and breakeven years across multiple scenarios (optimistic CRQC timeline, pessimistic timeline, and current trajectory). This gives a probability distribution over when HNDL attacks become economically viable.

Architecture: What This Exercise Tests

```
graph LR
    subgraph "Threat Simulator (threat_simulator.py)"
        A[CRQC arrival distribution<br/>P(CRQC by year Y)] --> B[Monte
Carlo Sampler]
        C[Storage cost trajectory<br/>$/GB/year declining] --> B
        D[Quantum compute cost<br/>declining with qubit scale] --> B
        B --> E[N=10,000 simulations]
    end
    subgraph "ThreatSimResult"
        E --> F[breakeven_year distribution]
        E --> G[profitable_data_types]
        E --> H[confidence interval]
    end
    subgraph "API"
        F & G & H --> I["GET /v1/quantum/{scan_id}/hndl<br/>
(timeline_details embedded)"]
    end
```

Step 1: View timeline simulation results for a critical device

```
# Full HNDL result for the highest-risk device includes timeline
        simulation
curl -s "http://localhost:8100/v1/quantum/$SCAN_ID/hndl?
        ip=172.30.0.10" \
```

```
  -H "Authorization: Bearer $TOKEN" \
  | jq '.data.risks[0] | {
      device_ip,
      risk_score,
      risk_category,
      quantum_arrival_years,
      data_shelf_life_years,
      storage_cost_per_gb,
      timeline_details
    }'
```

What you should see:

```
{
  "device_ip": "172.30.0.10",
  "risk_score": 82.5,
  "risk_category": "critical",
  "quantum_arrival_years": 7.0,
  "data_shelf_life_years": 15,
  "storage_cost_per_gb": 0.02,
  "timeline_details": {
    "harvest_window_years": 7.0,
    "data_still_sensitive_at_crqc": true,
    "economic_viability": "highly_profitable",
    "required_storage_tb": 0.12,
    "storage_cost_usd": 2.46
  }
}
```

Step 2: Compare timeline details across risk categories

```
# Get timeline details for one device from each risk category
for CAT in critical warning safe; do
  IP=$(curl -s http://localhost:8100/v1/quantum/$SCAN_ID/hndl \
    -H "Authorization: Bearer $TOKEN" \
    | jq -r --arg c "$CAT" '[.data.risks[] | select(.risk_category ==
        $c)] | .[0].device_ip')
  echo "--- Category: $CAT (IP: $IP) ---"
  curl -s "http://localhost:8100/v1/quantum/$SCAN_ID/hndl?ip=$IP" \
    -H "Authorization: Bearer $TOKEN" \
    | jq '.data.risks[0] | {risk_score, data_shelf_life_years,
        quantum_arrival_years,
          economic_viability: .timeline_details.economic_viability,
          storage_cost_usd: .timeline_details.storage_cost_usd}'
done
```

What you should see:

```
--- Category: critical (IP: 172.30.0.10) ---
{ "risk_score": 82.5, "data_shelf_life_years": 15,
```

```
"quantum_arrival_years": 7.0,
  "economic_viability": "highly_profitable", "storage_cost_usd": 2.46
}
--- Category: warning (IP: 172.30.0.50) ---
{ "risk_score": 45.0, "data_shelf_life_years": 5,
"quantum_arrival_years": 7.0,
  "economic_viability": "marginal", "storage_cost_usd": 0.42 }
--- Category: safe (IP: 172.30.0.22) ---
{ "risk_score": 15.0, "data_shelf_life_years": 1,
"quantum_arrival_years": 7.0,
  "economic_viability": "not_profitable", "storage_cost_usd": 0.02 }
```

Step 3: Analyse the HNDL economic model

```
# Aggregate economic viability across all devices
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/hndl \
  -H "Authorization: Bearer $TOKEN" \
  | jq '[.data.risks[].timeline_details.economic_viability] |
        group_by(.) | map({viability: .[0], count: length})'

# Total storage cost if an adversary harvests ALL vulnerable devices
curl -s http://localhost:8100/v1/quantum/$SCAN_ID/hndl \
  -H "Authorization: Bearer $TOKEN" \
  | jq '([.data.risks[] | select(.risk_category == "critical") |
        .timeline_details.storage_cost_usd // 0] | add) as $total |
       {total_harvest_cost_usd: ($total | round),
        critical_device_count: [.data.risks[] | select(.risk_category
        == "critical")] | length}'
```

What you should see:

```
[
  { "viability": "highly_profitable", "count": 12 },
  { "viability": "marginal",          "count": 5  },
  { "viability": "not_profitable",    "count": 3  }
]

{ "total_harvest_cost_usd": 30, "critical_device_count": 12 }
```

The total cost to harvest all critical devices is $30 -- making HNDL attacks accessible to any motivated adversary with minimal budget.

> **Troubleshooting:** If `timeline_details` is an empty object `{}` for all devices, the HNDL engine did not run the timeline model. This occurs when `data_shelf_life_years: 0` (device type not recognised). Check: `curl -s http://localhost:8100/v1/quantum/$SCAN_ID/hndl -H "Authorization: Bearer $TOKEN" | jq '[.data.risks[].data_shelf_life_years] | unique'` -- all zeros indicates the device type classifier did not map shelf life values.

Questions:

1. The Monte Carlo simulation uses `quantum_arrival_years: 7.0` as the median CRQC arrival estimate. What happens to the HNDL risk score if this estimate changes to 5 years (optimistic for adversaries) or 12 years (pessimistic)? Recalculate the `data_still_sensitive_at_crqc` flag for a device with `data_shelf_life_years: 10` under both scenarios. *(See Slides 098-100 for CRQC arrival probability distributions)*
2. The total cost to harvest all critical-rated devices is $30. This extreme low cost is the fundamental driver of the HNDL threat. Compare this to the cost of traditional data theft methods (purchasing stolen credentials, exploiting web vulnerabilities). Why does HNDL represent a qualitatively different threat model rather than just a different exploitation technique? *(See Slide 101 for threat economics comparison)*
3. The `economic_viability` field has three values: `highly_profitable`, `marginal`, and `not_profitable`. Design a decision rule an adversary would use to decide which devices to prioritise for HNDL harvesting. Consider: data sensitivity, harvest window length, storage cost, and probability of decryption success. *(See Slides 102-104 for adversarial decision modelling)*
4. Review all 10 exercises in this lab. Build a one-page quantum readiness action plan for a 20-device IoT network with a $10,000 annual security budget. Include: (1) immediate actions (this quarter), (2) short-term actions (6 months), (3) long-term migration (2 years). Reference specific API responses and metrics from your exercises to justify each priority. *(See Slide 105 for the complete quantum migration framework)*

Cleanup

```
# No persistent files created in this lab
# Variables can be cleared if desired:
unset TOKEN SCAN_ID DEVICE_IP MIGRATION_COST RISK_REDUCTION
```

If you are finished with the lab environment:

```
docker compose down -v    # Stop containers and remove data
```

Troubleshooting Reference

Authentication and Setup

Symptom	Cause	Fix
`Token: null` after login	User account does not exist	Create account via the registration endpoint or reset the la

Symptom	Cause	Fix
`Scan ID: null`	No completed scans	Run a scan: `python scan_report.py 172.30.0.0/24`
`401 Unauthorized` on any request	Token expired (30 min TTL)	Re-login: `TOKEN=$(curl -s -X POST http://localhost:8100/v1/auth/login -H "Conte application/json" -d '{"email":"student@example.com","password":"S | jq -r '.access_token')`

Phase 7 API Issues

Symptom	Cause	Fix
`"Quantum assessment not found"` (404)	Scan did not run Phase 7	Re-run scan with quantum phase enabled
`overall_score: null`	No TLS services discovered	Verify HTTPS devices exist: `curl -s http://localhost:8100/v1/analytics/grap -H "Authorization: Bearer $TOKEN" | jq '[.data.nodes[] | select(.metadata.port | length'`
`count: 0` on `/hndl`	No HNDL assessments persisted	Check `device_count` on scorecard; retry after ful
`timeline_details: {}`	Device type not mapped to shelf-life	IoT Sim uses type inference; verify device types v identified
`exposure_pct: 0.0` on HYDRA	Quantum assessments list empty	Run a new scan to repopulate quantum assessmen
`"Migration plan not found"` (404)	Planner did not run (no vulnerable devices)	Verify `vulnerable_count > 0` in scorecard
`total_cost: 0.0`	All devices scored as `safe`	Likely no TLS endpoints; migration planner only vulnerable devices

jq Errors

Symptom	Cause	Fix
`Cannot iterate over null`	Field is null instead of empty array	Add null coalescing: `.data.risks // []`

Symptom	Cause	Fix
`null (null) has no keys`	Accessing nested field on null object	Use `// {}` fallback: `.timeline_details // {}`

Network and Container Issues

Symptom	Cause	Fix
Connection refused on port 8100	API container not running	`docker compose up -d` and wait 30 seconds
Scan stuck at `in_progress`	Background scan still running	Wait; check status with `curl -s http://localhost:8100/v1/scanning/smart-scan/history?limit=1 -H "Authorization: Bearer $TOKEN" | jq '.scans[0].status'`

Slide Reference Index

Exercise	Topic	Slides
Setup	Phase 7 overview and quantum timeline	001-009
1	Quantum readiness scorecard	010-022 (esp. 018-021)
2	Cipher suite vulnerability analysis	023-035 (esp. 025-031)
3	HNDL risk profiling	036-048 (esp. 040-044)
4	Certificate lifecycle analysis	049-058 (esp. 052-057)
5	Cryptographic agility testing	059-068 (esp. 062-066)
6	Migration plan generation	069-075 (esp. 071-074)
7	TCO breakdown and ROI	076-080 (esp. 077-079)
8	PQ validation and hybrid chains	081-088 (esp. 084-087)
9	HYDRA stream and heatmap	089-096 (esp. 091-094)
10	HNDL timeline simulation	097-105 (esp. 098-104)
--	API design and endpoints	081-086
--	Dashboard quantum views	087-094
--	Testing strategy	106-111